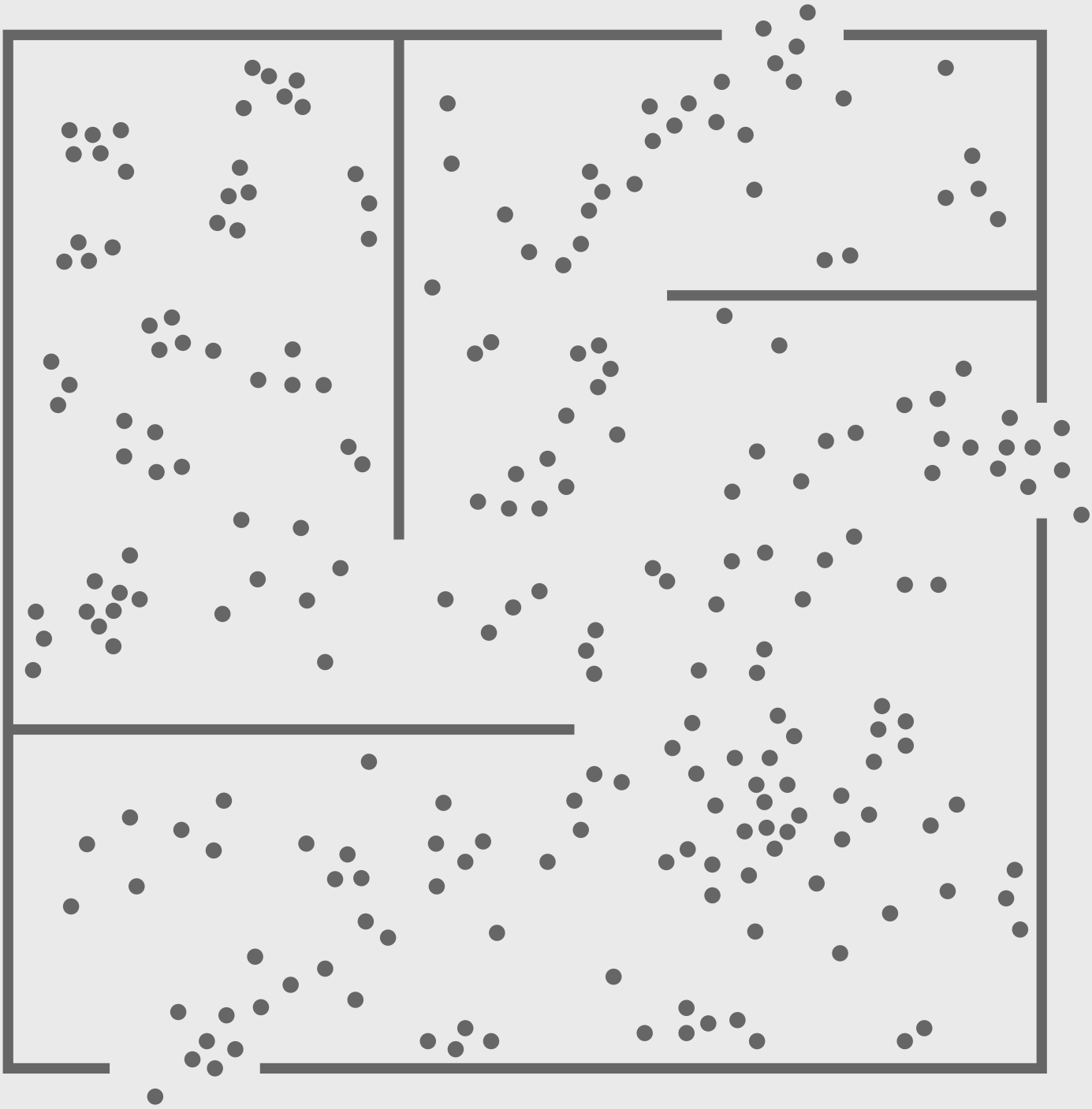
PHD THESIS

A MULTISENSOR ARCHITECTURE FOR GEOGRAPHICAL POSITIONING IN INDOOR ENVIRONMENTS



ANTONIO JESÚS RUIZ RUIZ

2014

UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

A multisensor architecture for geographical positioning in indoor environments.

Diseño de una arquitectura de localización geográfica para entornos de interior basada en información multisensor

D. Antonio Jesús Ruiz Ruiz

2014

UNIVERSITY OF MURCIA

FACULTY OF COMPUTER SCIENCE

Department of Computer Engineering

A multisensor architecture for geographical positioning in indoor environments.

D. Antonio Jesús Ruiz Ruiz

Supervised by:

D. Óscar Cánovas Reverte
D. Pedro E. López-de-Teruel Alcolea

2014

Abstract

In this thesis we present the architecture of a localization system that uses data from multiple sensors available in commodity smartphones. Our main aim is to provide a solution able to support accurate location-based services, such as augmented reality applications, pursuing a good balance between accuracy and performance. We present the architecture which encompasses the overall system proposed.

First we focus on the analysis of location fingerprinting techniques based on IEEE 802.11, feasible to determine the position of a device with a few meters of estimation error. Several refinements, based on the integration of additional sensors such as the camera or the inertial, are introduced to improve the efficiency of our solution. Using scale invariant features extracted from images we provide a solution for scene recognition that clearly improves the reliability of our result. Moreover, we take a step forward in the image analysis by including visual structure from motion techniques. It allows us to run off-line 3D reconstructions of the environment, and applying image resection techniques, we are able to provide precise estimations of both the 3D position and rotation of the device, obtaining an accuracy around 15 cm of error.

Our multisensor solution works in two different stages. We first obtain a coarse-grained estimation based on WiFi signals, digital compass, and built-in accelerometer, making use of fingerprinting methods, probabilistic techniques, and motion estimators. Then, using the images captured by the camera, we carry out the image analysis focusing on the subset of the 3D model spatially delimited by the previously obtained coarse estimation.

Because of the difficulties found to build accurate 3D models in large and repetitive environments, our proposal makes use of state-of-the-art IMU data processing techniques during the training phase, in order to reliably generate 3D representations of the targeted environment. This process solves typical scalability issues related to visually repetitive structures in large indoor scenarios. The fact of getting high accurate 3D representations of the testbed scenario improves the efficiency of camera resection techniques, reducing the estimation error to 5 cm, with response times below 250 ms. The set of techniques presented supports a wide range of location-based applications, from those requiring a coarse estimation to those with high accuracy requirements.

Resumen

Introducción

Los avances tecnológicos que han experimentado los dispositivos móviles durante los últimos años han favorecido su integración dentro de las tareas del día a día de las personas. Este hecho ha permitido mejorar la experiencia de usuario a la hora de interaccionar con el medio que nos rodea de diversas maneras. En la rama de la ingeniería informática, esta capacidad de interacción ha sido definida como "Computación Ubicua", la cual ha evolucionado de forma notable en gran medida gracias al desarrollo de dichos dispositivos móviles.

Cuando hablamos de dispositivos móviles nos referimos principalmente a los smartphones, cuyo uso es muy extendido hoy en día. Estos dispositivos han sido dotados con una amplia variedad de sensores que nos permiten explorar el entorno. Esta capacidad, junto con las mejoras de potencia de computo que ofrecen, nos permite realizar labores de captura y análisis de información contextual que hace unos años parecía imposible.

El concepto de "ubicuidad" está estrechamente relacionado con el término "localización", el cual es el principal objetivo de esta tesis. Como hemos comentado anteriormente, la computación ubicua se basa en la mejora de la integración del usuario dentro del medio que le rodea haciendo uso de las nuevas tecnologías. De aquí podemos extraer que saber la posición en la que se encuentra una persona es clave para favorecer dicha integración. Es en este punto donde hemos detectado algunas carencias que vamos a intentar resolver en esta tesis.

Obtener la posición de una persona es un proceso relativamente sencillo cuando dicha persona se encuentra en un entorno de exteriores. Técnicas muy consolidadas y de uso muy común como es el caso de GPS, ayudan a resolver el problema de forma eficiente. Sin embargo, el problema de localizar a una persona cuando se encuentra en el interior de un edificio no es algo trivial de resolver, quizás por eso aún no se ha encontrado la solución óptima.

Si pensamos en cuál es el comportamiento normal de las personas, podríamos afirmar sin lugar a dudas que la mayor parte de nuestro tiempo la pasamos dentro de algún edificio (hogar, oficina, centros comerciales, etc.). Además, conforme la tecnología ha ido evolucionando y los smartphones se han hecho imprescindibles para las personas, han ido apareciendo nuevos servicios dependientes de la localización geográfica del dispositivo (marketing, navegación, entretenimiento, etc.). Por ello surge la gran necesidad de ser capaces de proveer tal información de posicionamiento a aquellos servicios que lo

requieran, sea cual sea el entorno en el que nos encontremos, incluido dentro de un edificio.

Actualmente, existe una gran variedad de propuestas que se centran en resolver el problema de la localización para interiores, pero la mayoría de ellas han sido creadas para cubrir una necesidad específica, no siendo capaces de adaptarse a las distintas necesidades que puedan surgir. Se debe ofrecer distintos niveles de precisión en las estimaciones, atendiendo a la heterogeneidad en cuanto a las características de los dispositivos con los que nos podemos encontrar. Por ello, el principal objetivo general que nos marcamos al inicio de esta tesis fue el de diseñar una arquitectura que permitiera desarrollar un servicio de localización preciso, manejando información obtenida por diferentes sensores integrados en los smartphones. Nuestra propuesta multisensor será capaz de adaptarse a las necesidades específicas de cada aplicación, de acuerdo a la precisión requerida y a las restricciones de sensores que puedan existir.

Técnicas de localización en interiores

Durante las últimas tres décadas, dentro del campo de investigación de la localización en interiores se ha hecho uso de una amplia variedad de tecnologías gracias a la cuales se han realizado interesantes aportaciones. En el *Capítulo 2*, hemos hecho un amplio resumen de las diferentes propuestas que hemos ido analizando durante nuestro periodo de investigación. Puesto que vamos a trabajar con smartphones, pondremos especial atención en aquellas técnicas basadas en tecnologías disponibles para dichos dispositivos, tales como señales 802.11, imágenes e información obtenida por sensores inerciales.

Las líneas de investigación en el campo de la localización en interiores se han centrado principalmente en el estudio de la tecnología 802.11. Se han llevado a cabo numerosos proyectos en los que se demuestra la utilidad de esta tecnología para desarrollar sistemas de localización que ofrecen ciertas garantías tanto en precisión como en fiabilidad. Las principales ventajas a destacar de esta tecnología son: su amplia implantación en casi todos los edificios tanto privados como públicos, en grandes superficies como hipermercados, en centros comerciales o aeropuertos; la posibilidad de utilizar una misma infraestructura de red tanto para conexión a Internet como para llevar a cabo el proceso de localización; el bajo coste de la misma ya que tanto los dispositivos como el servicio de conexión a la red no tienen precios elevados; la casi absoluta implantación de esta tecnología en los dispositivos móviles.

Otras técnicas de localización que discutiremos se centran en el análisis de la información de contexto capturada por las cámaras integradas prácticamente en todos los dispositivos

móviles. Por lo general, las técnicas propuestas que hacen uso de imágenes para proveer información de posicionamiento suelen ofrecer muy buenos resultados de precisión. Sin embargo, el principal inconveniente es su alto coste computacional. Finalmente, también veremos una serie de técnicas que hacen uso de los datos ofrecidos por los sensores inerciales para estimar los movimientos realizados por el dispositivo.

Hoy en día, los principales motivos que despiertan el interés de los investigadores por innovar dentro de este campo no van tanto en la dirección de intentar mejorar la precisión obtenida por sistemas ya propuestos, sino que se centran en la integración de sensores dentro del sistema de localización que suponga un valor añadido para el mismo. En este sentido, se han propuesto trabajos relacionados con el uso de otros sensores disponibles en los dispositivos actuales, donde se trabaja en la fusión de la información obtenida mediante el hardware 802.11 con la obtenida por sensores de sonido, acelerómetros, giróscopios o cámaras, entre otros, con la finalidad de aportar nuevos datos que ayuden a llevar a cabo el proceso de localización de un dispositivo de una manera eficiente.

Nuestra propuesta multisensor

Hasta hoy la mayoría de las propuestas multisensor desarrolladas se centraban en cubrir las necesidades específicas del problema al que se enfrentaban. En nuestro caso hemos diseñado un sistema que se vale de dicha multimodalidad con el fin de dar soporte a una amplia variedad de aplicaciones basadas en localización, intentando cubrir distintas carencias que hemos observado. En ese sentido, los objetivos que nos marcamos al inicio de esta tesis fueron los siguientes:

- Diseñar una solución capaz de adaptarse a las necesidades específicas de cada escenario de uso. Por ello nuestro principal objetivo es crear una solución que se adapte a los requisitos de precisión impuestos por las aplicaciones y al mismo tiempo se adapte a las características y capacidades de los dispositivos a localizar. Gracias a ello seremos capaces de ofrecer información de localización tanto a aquellas aplicaciones que requieran una precisión de grano grueso (por ejemplo, a nivel de habitación, planta de edificio, etc.) como a aquellas otras que requieren una mayor precisión (por ejemplo, aplicaciones de realidad aumentada). Este objetivo debe ir acompañado de la necesidad de intentar minimizar el tiempo de respuesta.
- Nuestro segundo objetivo se centraba en mejorar la fase de entrenamiento. Los sistemas de localización basados en técnicas de mapas de huellas de sensores (como es nuestro caso), requieren una fase previa de entrenamiento o aprendizaje,

necesaria para crear los modelos que nos ayudarán a realizar las estimaciones de localización. En nuestro sistema, la necesidad de crear modelos 3D del entorno para poder hacer estimaciones precisas supone una dificultad añadida al proceso de entrenamiento. La creación de estos modelos requiere la supervisión de un operador para evitar aquellas situaciones de error que se puedan dar y que producirían como resultado modelos que no se ajustarían a la realidad. Este objetivo se centra en la definición de una metodología que permita llevar a cabo esta fase de entrenamiento y generación de modelos de una forma automática y no supervisada.

- Finalmente nuestras propuestas se engloban dentro del objetivo global de esta tesis, el de ofrecer una arquitectura escalable, extensible y modular que permita diseñar diferentes tipos de sistemas de localización. Dicha arquitectura deberá ser capaz de manejar la multimodalidad ofrecida por los smartphones, de forma que se pueda integrar la información extraída de los distintos sensores disponibles.

Como se verá a lo largo del documento, el proceso de desarrollo de nuestro sistema de localización se divide en tres fases claramente diferenciadas.

En primer lugar se llevó a cabo una fase de análisis de distintas técnicas de localización basadas en mapas de huellas, *Capítulo 3*. Como indicamos anteriormente, la principal tecnología empleada fue 802.11. Esta primera fase nos permitió llevar a cabo la integración de diferentes propuestas ya existentes, desarrollando una solución que ofrece buenos resultados en cuanto a precisión y rendimiento. Los resultados obtenidos estaban en el entorno de los 3 metros de error medio de estimación, consiguiendo dar respuesta a las peticiones en apenas unos pocos milisegundos.

Sin embargo, la precisión obtenida no era suficiente para alcanzar nuestro objetivo de dar soporte a aplicaciones con altas exigencias en ese aspecto. Por ello iniciamos una segunda fase de investigación y desarrollo en la cual se integró la cámara como segundo sensor dentro de nuestro sistema de localización. Como se detalla en el capítulo correspondiente, *Capítulo 4*, el análisis de imágenes se llevó a cabo en dos fases.

En la primera de ellas, introducimos el uso de imágenes con el fin de mejorar la fiabilidad de las estimaciones realizadas mediante el análisis de las señales WiFi. Esta primera versión multisensor de nuestro sistema hace uso de mapas de huellas de imágenes para realizar estimaciones de posición mediante el análisis de las imágenes tomadas durante la fase de localización. Este análisis se ha llevado a cabo usando un algoritmo de extracción de características distintivas de imágenes en escala de grises, y que puede ser utilizado para reconocer la misma característica entre diferentes vistas de un mismo objeto o escenas. Esto nos permite llevar a cabo el reconocimiento de escenas con una alta probabilidad de acierto, consiguiendo un nivel de precisión en nuestras estimaciones de

localización proporcional al tamaño de las celdas en las que se divide nuestro espacio de búsqueda. El principal problema de la integración de imágenes es el elevado coste computacional de realizar la búsqueda en grandes bases de datos de imágenes de entrenamiento. Para ello, nuestra primera versión multisensor se basa en realizar una estimación de grano grueso inicial de la posición del dispositivo, haciendo uso de las señales WiFi. Estas estimaciones nos ayudan a la hora de acotar la búsqueda de imágenes, mejorando de forma considerable la eficiencia de nuestra solución, al tiempo que favorecen la escalabilidad de la misma.

En la segunda fase del análisis de imágenes hacemos uso de técnicas avanzadas de visión por computador, como por ejemplo técnicas de resección de cámara. Esto nos permite realizar estimaciones precisas de los seis grados de libertad de la posición de una cámara, es decir, los tres grados de libertad que indican su posición en el espacio 3D y los tres grados de libertad que indican la orientación, rotación e inclinación de la cámara. El uso de estas técnicas nos permitió mejorar la precisión de nuestras estimaciones, reduciendo el error cometido a unos 15 centímetros de media. Estas técnicas requieren disponer de modelos 3D a escala del entorno para establecer las correspondientes correspondencias entre los puntos 2D situados en la imagen con su posición 3D en el mundo real. La gen-eración de dichos modelos 3D es una tarea costosa, no solo en términos de cómputo ya que se debe ejecutar de forma off-line, sino porque dicha generación puede verse afectada de forma negativa por la existencia de elementos similares en diferentes partes del escenario. El tamaño del escenario a modelar también puede ser un problema importante en este caso. Estos problemas impiden un proceso de construcción limpio, y requieren la supervisión de un operador para solucionar los posibles casos de inconsistencia que puedan darse.

Por estas razones, otra de nuestras aportaciones es la propuesta de una solución diseñada para apoyar la fase de entrenamiento en aquellos entornos en los que se requiera construir un modelo 3D del mismo, *Capítulo 5*. Esta solución es capaz de minimizar los problemas previamente comentados, mejorando la fiabilidad del proceso de reconstrucción y por lo tanto la precisión de los modelos obtenidos. Para ello hemos realizado un análisis en de los datos facilitados por distintos sensores tales como el acelerómetro, la brújula digital o el giroscopio, siendo capaces de estimar de forma aproximada la posición del operador durante el proceso de entrenamiento. Esto nos permite etiquetar las muestras de todos los sensores con la posición en la que son obtenidos, lo que supone una importante ventaja a la hora de llevar a cabo el proceso de reconstrucción de los modelos 3D. Este método de generación de modelos 3D hace que éstos se ajusten mejor a la realidad, teniendo una notable influencia en la precisión obtenida durante la fase de localización, dejando el error medio de estimación en torno a los 5 centímetros.

Finalmente, con el fin de cubrir el objetivo principal de esta tesis, en el *Capítulo 6* presentamos la arquitectura que engloba nuestro sistema de localización. Dicha arquitectura ha sido diseñada de forma modular, con el objetivo de poder adapatarla a diferentes escenarios de aplicación. La funcionalidad que debe integrar un sistema de localización ha sido divida en diferentes planos que definen las responsabilidades de cada módulo diseñado.

Conclusiones

A lo largo de esta tesis se proponen diferentes soluciones para alcanzar los objetivos que nos marcamos al inicio. Se propone una arquitectura multisensor extensible, escalabe y adaptable que ofrece los mecanismos necesarios para el desarrollo de sistemas de localización. Se hace uso de diferentes técnicas basadas en el análisis de señales WiFi para proporcionar un servicio básico de localización. De esta manera somos capaces de realizar una estimación aproximada, en el rango de unos 3 metros de error, de la posición de un dispositivo con una relativa fiabilidad. El uso de imágenes nos permite mejorar la fiabilidad de las estimaciones realizadas, siendo capaces de obtener la posición de un dispositivo con 5 centímetros de error. El análisis de los datos de los sensores inerciales obtenidos durante la fase de entrenamiento permite estimar la ruta seguida por los operadores. El camino inferido posibilita el geo-etiquetado automático de las mediciones obtenidas por otros sensores (WiFi y cámara), y facilita la creación automática de mapas de huellas de señales de radio y de imágenes. Como demostraremos, este proceso da lugar a reconstrucciones más confiables de los mapas 3D de las escenas.

El tiempo de respuesta conseguido ronda los 250 milisegundos. Somos conscientes de que esta tasa de refresco es todavía insuficiente para dar soporte a aplicaciones con tasas de entrada de imágenes del entorno de los 25 frames por segundo. Sin embargo podemos pensar en muchas aplicaciones móviles que no requieren tal frecuencia de actualización de información de posicionamiento.

Al finalizar esta tesis somos conscientes de que aún queda mucho trabajo que hacer. Por un lado, las propuestas que hemos realizado son susceptibles de mejora en varios aspectos, como por ejemplo intentar reducir el tiempo de respuesta, o integrar la extracción de las características de las imágenes en los propios smartphones. Por otro lado, una vía futura de desarrollo sería la integración de nueva funcionalidad, como podría ser la fusión de datos inerciales durante la fase de localización, o la integración de otros sensores no considerados hasta ahora por nuestra parte. Por último, un elemento fundamental a tratar será la integración de las medidas de seguridad necesarias para asegurar la confidencialidad de los datos manejados y para mantener el anonimato de los usuarios.

A pesar de esto, creemos que hemos dado un paso importante para demostrar que la integración multisensor es el camino a seguir de cara a desarrollar soluciones eficientes de localización para interiores.

A Natalia y a Marta, por hacer especial cada día de mi vida junto a vosotras.

Acknowledgements

En primer lugar quiero dar las gracias a mis directores Óscar y Pedro, por todo el apoyo que me han ofrecido durante estos años de trabajo y por el gran esfuezo que han hecho para que esta tesis llegara a buen puerto. Gracias por haber tenido siempre tiempo para mí, por haberme dado la oportunidad de conocer vuestro perfil docente y personal, por los todos buenos momentos que hemos pasado y que con el tiempo echaré de menos, y por haberme acompañado en esos viajes que nos han hecho recorrer medio mundo.

Gracias Óscar por confiar en mí desde el inicio, por introducirme a fondo en el mundo de la investigación y por haberme dado la opotunidad de llevar a cabo este trabajo. Gracias por haberme guiado durante estos años, por tu paciencia y por haber tenido siempre buenas palabras hacia mí. Pedro, aunque te subiste al barco cuando este ya había zarpado, tengo que darte las gracias por haber hecho el esfuerzo de nadar desde la orilla hasta unirte a este proyecto. Tus pizarras interminables, dignas de ser fotografiadas, siempre quedarán para el recuerdo.

A mi familia, abuelos, hermana, y en especial a mis padres, porque siempre han estado ahí. Gracias por gran esfuerzo y sacrificio que habéis realizado a lo largo de todos estos años con el fin de permitirme alcanzar todos mis objetivos. Siempre habéis sido el mejor ejemplo a seguir, en todos los sentidos. Durante los años de doctorando, aunque desde la distancia, siempre habéis conseguido trasladarme vuestro apoyo, mis preocupaciones eran vuestras preocupaciones y ahora este logro también quiero que lo sintáis como vuestro.

A mis compañeros, por el buen ambiente que siempre ha existido entre nosotros y que sin duda ha hecho más amenas las horas dentro del laboratorio. Gracias a Ana, por ser la primera en recibirme en el departamento, por su apoyo y por esas largas charlas compartiendo todo aquello que nos preocupaba. Gracias por haber inventado el tissue-tennis, que tantos momentos divertidos nos ha hecho pasar junto con Jesús. Gracias a Jordi, por estar siempre ahí para ayudarme a resolver cualquier problema, y a Alejandro por su compañía incondicional a la hora de salir a almorzar. Gracias a todos mis compañeros Danipg, Toni, Rubén, Chema, Dani, Juanma, Alberto, Ginés, Jose Luis, Epi, Jesús, Pedro y Andrés, por todos los buenos momentos que pasamos juntos cuando el departamento aún estaba lleno de becarios. Gracias también a aquellos que habéis estado colaborando en este trabajo, Rubén, Juan y Alejandro, por vuestra disponibilidad y la ayuda prestada.

Quiero dar las gracias al resto de profesores del departamento DITEC, por su amabilidad y por su cercanía tanto conmigo como con mis compañeros. En especial quiero dar las

gracias a Félix, con el que he tenido la oportunidad de colaborar, por el interés que siempre ha mostrado en mi trabajo.

I would also want to thank those that made my stay in Aarhus unforgettable. I want to especially thank to Mikkel, for believing in me and giving me the opportunity of living that incredible experience. Thanks to Henrik for his support and dedication. I am also grateful to Thor, Allan, Mads, Nervo and the rest of guys that made me feel like home.

Finalmente quiero dar las gracias a Natalia, mi mujer, ya que sin su ayuda no habría sido posible llevar a cabo esta tesis. Gracias por estar ahí en todo momento, por haber sabido darme ánimos en los momentos más complicados, y por tu apoyo incondicional en todos los aspectos de mi vida. Gracias por tu comprensión y por haberme permitido lograr este objetivo. Durante estos años me has dado lo más grande que jamás nos ha podido pasar, a Marta, que sin duda ha hecho esta aventura más complicada pero a la vez más gratificante. Ella es mi principal fuente de energía e inspiración, y la que cada día renueva la ilusión para seguir adelante en el cumplimiento de nuestros objetivos. Gracias de corazón.

Contents

List of Figures

List of Tables

Abbreviations

A-AOA	**A**zimuth **A**ngle **O**f **A**rrival
A-GPS	**A**ssisted **G**lobal **P**ositioning **S**ystem
A-SIFT	**A**ffine **S**cale **I**nvariant **F**eature **T**ransform
ANN	**A**pproximate **N**earest **N**eighbor
AOA	**A**ngle **O**f **A**rrival
AP	**A**ccess **P**oint
API	**A**pplication **P**rogramming **I**nterface
AR	**A**ugmented **R**eality
CCTV	**C**losed **C**ircuit **T**ele**V**ision
CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
DLT	**D**irect **L**inear **T**ransform
DR	**D**ead **R**eckoning
E-AOA	**E**levation **A**ngle **O**f **A**rrival
ED	**E**arly **D**etention
ENN	**E**xact **N**earest **N**eighbor
EP	**E**ntry **P**oint
FAF	**F**loor **A**ttenuation **F**actor
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
GIS	**G**eographical **I**nformation **S**ystem
GLONASS	**GLO**balnaya **NA**vigatsionnaya **S**putnikovaya **S**istema
GLSL	open**GL** **S**hading **L**anguage
GNU	**G**nus's **N**ot **U**nix
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
GSM	**G**lobal **S**ystem for **M**obile communications

HLF	**H**yperbolic **L**ocation **F**ingerprinting
HMM	**H**idden **M**arkov **M**odel
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IMU	**I**nertial **M**easurement **U**nit
INS	**I**ndoor **N**avigation **S**ystem
IT	**I**nformation **T**echnology
IT	**I**ncremental **T**riangulation
JSON	**J**ava **S**cript **O**bject **N**otation
LBS	**L**ocation **B**ased **S**ervice
LOS	**L**ine **O**f **S**ight
MAC	**M**edium **A**ccess **C**ontrol
MLP	**M**ulti **L**ayer **P**erceptron
NLOS	**N**on **L**ine **O**f **S**ight
OS	**O**perating **S**ystem
P3P	**P**erspective **3** **P**oints
POI	**P**oing **O**f **I**nterest
PRL	**P**ath **R**estricted **L**ocation
RANSAC	**RAN**dom **SA**mple **C**onsensus
REST	**RE**presentational **S**tate **T**ransfer
RF	**R**adio **F**requency
RFID	**R**adio **F**requency **ID**entification
RGB	**R**ed **G**reen **B**lue
RNN	**R**ecurrent **N**eural **N**etwork
RSS	**R**eceived **S**ignal **S**trength
RSSI	**R**eceived **S**ignal **S**trength **I**ndicator
SfM	**S**tructure **f**rom **M**otion
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SME	**S**pace **M**odel **E**ntity
SNR	**S**ignal to **N**oise **R**atio
SP	**S**pace **M**odel
SSL	**S**ecure **S**ocket **L**ayer

SURF	**S**peeded **U**p **R**obust **F**eatures
TDOA	**T**ime **D**ifference **O**f **A**rrival
TIX	**T**riangular **I**nterpolation and e**X**trapolation
TLS	**T**ransport **L**ayer **S**ecurity
TOA	**T**ime **O**f **A**rrival
UC	**U**biquitous **C**omputing
UWB	**U**ltra **W**ide **B**and
WAF	**W**all **A**ttenuation **F**actor
WiFi	**W**ireless **F**idelity
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WOL	**W**eb **O**ntoloty **L**anguage
ZUPT	**Z**ero-velocity **UP**da**T**e

Chapter 1

Introduction

The concept of *ubiquitous computing* (UC) was coined by Weiser [184] in the late 80s, defining it as a paradigm centered on the idea of integrating devices within the environment, in such a way that they offer an optimal support to human daily life activities. According to Weiser's perspective, a computer should be used to provide us with an artificial extension of the reality we live in and interact with.

During the last three decades computing devices have been made more compact while at the same time more efficient. The continuous integration of a wide variety of sensors has allowed the full interaction of these devices with the surrounding context, making them able to explore the environment and realizing about what is taking place. Clear examples of such devices are the smartphones and tablets, which nowadays count with an important set of sensors (like WiFi connectivity, light, acoustic, cameras, accelerometers, compasses, among other) which make them able to measure contextual information of a different nature. The widespread adoption of these devices, combined with the advances in the deployment of radio networks, make it feasible to keep them always connected, facilitating their cooperation and the communication with external services in order to accomplish difficult tasks.

In this thesis we focus on the use of smartphones as the key tool to scan the context, taking advantage of the diversity of resources they provide. The rapid evolution of these devices during the last years has been so intensive that, apart from the basic services they offer, people in general still ignore many of their possibilities. The term *smartphone* was coined in 1993, being first used to classify the IBM's Simon personal communicator

device. Throughout the remainder of the 1990's, various types of advanced mobile phones appeared on the market. Touch screens began to replace keyboards on many smart phones, while games and email functions became more sophisticated as new models were released. However it was not until the end of the last decade, with the incursion of the first version of the Apple's iPhone in 2007 or the first Android OS device, the HTC Dream in 2008, when the real explosion of the smartphones took place. During the past six years the increasing number of people having a smartphone (e.g. more 55% of Spain's population in 2013 according to statistics facilitated by Google in [13]) has made them a really *ubiquitous* device.

The concept of *ubiquity* is closely related to the concept of *localization*, which is the keystone of this thesis. It is based on the capacity of knowing the position of a specific device in order to provide information regarding its state, to offer the most appropriate services or functionality associated with the context in which it is currently located. In that sense, the ability of estimating the position of a device becomes essential in those scenarios where the services provided have a strong dependency on that information, like navigation, marketing, or gaming, to name a few.

As it is well known, the localization problem in outdoor environments was successfully solved by the GPS (*Global Positioning System*) [84], the GLONASS (*Global Orbiting Navigation Satellite System*) [85] or the still to be deployed GALILEO [132] technologies. Different localization solutions and devices have been developed based on these technologies, which have been integrated in several aspects of our life, such as car navigation, emergencies, and most recently in smartphones providing them with valuable information to offer a wide variety of location aware applications which support our daily activities.

Nevertheless, nowadays people spend most of their time in indoor environments. Studies like the one published in 2004 by the Eurostat (*Statistical Office of the European Communities*) [54] reflect the daily habits of people aged 20 to 74 in different UE countries. The statistics shown in Table 1.1 indicate that people spend around the 86% of their time inside buildings. This percentage includes the time spent at home, at the office, inside educational buildings, malls, stores or restaurants, among others. The remaining 14% refers to the time people spend outdoors, mainly activities like traveling (moving in a vehicle or walking) and other like doing sports. Since the habits of people in recent

years have not varied so much, these results demonstrated the necessity of providing localization solutions for indoor environments.

	Belgium	Germany	Estonia	Hungary	Slovenia	Finland	Sweden	Norway	Average
Home	67	65	67	70	69	64	61	63	**65,75**
Working place or school	13	14	18	15	16	14	17	18	**15,63**
Restaurant, cafe or pub	2	1	0	2	1	1	1	1	**1,13**
Other indoor places	2	4	3	4	2	5	4	4	**3,50**
Traveling (outdoor)	7	6	5	4	6	6	6	6	**5,75**
Other outdoor places	9	10	7	5	6	10	11	8	**8,25**

TABLE 1.1: Time distribution indoors and outdoors (%) during a work day for employed people in UE counties with ages between 20 to 74.

The remarkable success of GPS in the outdoor environment created a general consumer expectation that similar indoor accuracy should be available using the GPS system. However, the signals from GPS satellites are too weak to be reliable within buildings. Their structure and the elements they contain block the transmission of signals to be correctly received by the devices located inside, which prevents an appropriate functioning. Other approaches, such as GSM (*Global System for Mobile communications*) cell towers triangulation or assisted GPS (*A-GPS*), whether used alone or in combination, are not sufficiently precise to obtain accurate position estimations.

Determining location at indoor environments presents special engineering, social, and regulatory challenges. The wide scale of indoor environments and the variety of services provided require location accuracy ranged from coarse-grained estimations (room level) to fine-grained estimations (few centimeters of error). Sometimes we have to work within crowded or cluttered environments, making sensing difficult due to obstacles present. Additionally, intrinsic privacy concerns arise from the specter of *big brother* systems that monitor users constantly. Finally, commercial regulation limits the available technologies and thus the possibilities of developers to provide more accurate and deployable solutions.

1.1 Indoor localization overview

For the last 30 years, indoor positioning has become an important research topic. In its early stages, this unexplored field was a clear research objective not only for research groups of universities all around the world, but also for important IT companies. All of them have invested huge amounts of resources (human and economic) which resulted in the provision of a diversity of solutions to this problem. Among other alternatives,

most of them were based on the utilization of technologies commonly available in indoor scenarios, like WiFi, Bluetooth or RFID, and the analysis of additional information that can be collected by the sensors integrated in mobile devices, such as the images captured by the cameras, or the measurements collected from inertial sensors.

Among the vast number of proposals that have been made within this field we highlight those which marked a milestone in the evolution of these systems. The Active Badge [181] location system developed by the Olivetti Research Ltd, now AT&T Cambridge, was the first automated indoor location system. It used a diffuse infrared beacon embedded into the electronic badges which provided information about their position, obtaining room size accuracy. In 1997 AT&T researchers evolved this solution proposing the Active Bat [76] system, which provided a more accurate physical positioning using ultrasound time-of-flight lateration techniques to compute the coordinates of the mobile devices in relation to the known locations of ceiling-mounted units. Another alternative was the Cricket [145] system developed by the MIT in 2000. In this solution ceiling-mounted devices simultaneously transmitted radio packet and ultrasound pulses as a combined beacon. Though less accurate than previous systems, around 1 meter of estimation error, it provided other advantages regarding the privacy of users and the scalability of the system, since mobile devices were in charge of estimating their own location. The main disadvantages of all these techniques were the need for specific hardware and the deployment cost of having to use dedicated sensor networks.

Other key works in this field were the RADAR [24] positioning system and the EasyLiving [112] project, both developed by Microsoft research groups in 2000. The former is an indoor position tracking system which used the existing WLAN infrastructure to provide estimations around 3 meters of error. The latter was the first approximation to track people inside a room using the images captured by multiple cameras. In 2005 the Intel research labs proposed Placelab [117], a solution that integrates outdoor and indoor positioning, combining GSM cell tower transmissions and WiFi signals from access points throughout a city. Curiously, this technology was integrated in the early versions of the Apple's iPhone.

These works laid the foundations for the development of localization techniques for indoor environments. Thereafter, researchers all around the world have proposed a wide

variety of solutions using different sensor configurations, as we will describe in following chapters.

Nevertheless, the recent history of indoor positioning is clearly led by the most important worldwide technology companies, like Google, Apple, or Microsoft. They soon noticed the capacities of the indoor positioning based services, and in the last few years have maintained a hard competition in order to gain the leadership in this field.

In 2012 Google integrated indoor navigation within its *Google Maps* service for thousands of buildings in different countries, including office buildings, airports, shopping malls, and other public facilities. Nowadays Google is focused on improving its solution for the automatic generation of indoor maps with crowdsensing approaches, using the contextual information (mainly WiFi signals and images) observed by Android OS smartphones. Additionally, the Google's project *Tango* [10] offers a way to develop highly accurate indoor maps and renderings, a key requirement for indoor positioning systems requiring precise device position estimations. Their developers have created a smartphone which integrates a motion tracking and depth sensing camera which supports the automatic reconstruction of 3D maps of indoor spaces. Nowadays, this project is still in its early stages, but it could revolutionize indoor positioning and navigation in the coming years.

Meanwhile, Apple has also made important advances in the field of the indoor localization. During the last years Apple has collaborated with expert companies in the analysis of WiFi signals to identify the location of a device in indoor spaces, like WiFiSLAM and Wifarer, even acquiring the first one. With these strategic movements Apple gained an important role within the indoor positioning market. As a consequence Apple has been seeding commercial spaces with its own *iBeacon* technology [2] for months, making use of the sensors available in its iPhones and networks of WiFi beacons installed in the scenario. As in the case of Google, this project enables the company to map large indoor areas of stores, offices, event spaces and commercial buildings.

Finally, though having a higher experience than the previous ones, nowadays Microsoft is considered as the third largest company in the field of indoor positioning. As we have previously indicated, during the last decade Microsoft research division has dedicated an important amount of resources to the proliferation of indoor positioning technologies. Its most important location service, *Microsoft's Bing Maps*, enables another alternative that

counts with thousands of indoor maps of airports, shopping malls, and public buildings in North America, Europe, and Asia.

Due to the high availability of proposals and the diversity of services provided, new and more exciting applications have been developed to serve the mass consumer markets. Next, we look at some of the most common uses for indoor location information, which can be classified in these categories:

- *Locating people or assets indoors*: this is the most fundamental and valuable proposition for indoor location technology. Usually, these systems provide services like monitoring and tracking of devices. Examples of this type of systems are the commercial solutions provided by Ekahau [7] or the Aeroscout [1] solutions, which have been utilized in health care scenarios to improve the safety of senior citizens in residential centers, and integrated in manufacturing, mining and security environments, among other scenarios.

- *Navigation*: this category is closely related to the previous one, but requiring additional integration with information about the physical characteristics of the environment. Currently different solutions exist able to provide navigation information inside buildings, as for example the NaviKit service provided by Meridian [5] or the indoor navigation solution offered by Infsoft [4], both providing a software development kit for creating final applications. These solutions have been adopted for the guidance of disabled people inside buildings, to minimize the time required to reach a point of interest inside large buildings such as airports, or to guide people in emergency situations.

- *Gaming and entertainment*: the advances in visual computing and the opportunity to apply location information to combine the virtual and the real world create numerous and interesting new applications for people of different ages and demographics. Current work in augmented reality systems will form the basis for implementing applications that allow devices and users to interact in the real and virtual worlds simultaneously, as for example to develop augmented reality games like Ingress [3, 44], interactive guides for museums or augmented reality shopping applications, among others. Moreover, location data are used nowadays in social networking as an entertainment service. Social networking applications like

Facebook or Twitter use this information to offer services related to the current position of their users.

- *Analysis of group activities*: location technology can further assist users in the analysis of activities between people and things that are moving dynamically with respect to each other. This is an interesting topic, which involves the localization of groups of people who are known to be nearby, and the analysis of their behavior to extract additional knowledge useful to provide more personalized services. Interesting proposals have been made by Neils et al. in [106] and Kjærgaard et al. in [104], which open new opportunities for the development of additional services.

- *Marketing and personal advertising*: within this category we find those solutions that make use of indoor positioning services for context-aware advertising. The main goal is to provide alternatives that improve the way in which people receive information about products, events, or services they might be interested in. An example of such type of systems is the Sensewhere project [8], developed by a research group of the University of Edinburgh.

These examples briefly represent the evolution of indoor positioning techniques and current activity carried out by important companies and universities. Nowadays, Google seems to be the best positioned in this field, mainly supported by the notable leadership in the domain of the worldwide smartphone's market, in which Android OS devices move towards 60% of share, against the 35% of Apple as the most direct competitor, according to a recent study made by Tech-Thought company [169].

While the already mentioned usages and the developed solutions cover a vast spectrum of applications, the applicability for location information is limited only by our imagination. The report [14] made by ABI Research predicts that the ecosystem necessary for mass adoption of indoor location applications will be ready by 2016. According to this report, the constant evolution of indoor location and proximity technologies makes evident that they will play a key role in the future of mobile technology. Another study by Research and Markets [12] provides interesting information on the future development of indoor positioning systems. The prognostic for 2018 is that over 800 million of smartphones will actively use indoor location services for applications, becoming as standard as the GPS is today.

1.2 Motivations and goals of this thesis

Before determining the main research lines of this thesis, we carried out an in-depth analysis of the state of the art of the indoor localization systems. This has allowed us to be aware of its importance within the research community, and the implications of managing location information for current and future services. Apart from this, we found several lacks and opportunities that motivated the work carried out during the last four years.

In the initial stages of this thesis, early 2010, the advances in this research field were limited by different factors:

- The lack of facilities (sensors) in smartphones to scan the context.
- The computational limitations of these devices to accomplish intensive computing tasks.
- The monomodal nature of most of the available solutions. Multisensor approaches integrated the use of additional sensors just to accomplish a specific task, but usually not providing a real fusion of technologies.
- The requirement of specific deployments (such as RFID or ultrasounds) to obtain high accuracy estimations. Solutions based only on existing infrastructures (like WLAN networks) provided an accuracy around 3 meters of estimation error on average.
- Most of the solutions provided were designed with specific purpose only, which limited its extensibility and applicability in other scenarios.
- The difficulties to carry out an optimal acquisition of the information required to build the sensor models (fingerprint maps) that support the localization process.

However, the constant evolution of the smartphones led us to think that this progress would continue in following years, not only from the point of view of the improvement of their computational capacities, but also from the perspective of the integration of additional and more reliable sensors. Consequently, we took advantage of the multisensor nature of the smartphones, in order to improve the acquisition of context data and to

provide a location service supporting different granularity. We were aware that our contribution within the indoor localization field should not be addressed to find a new technique that definitely solved the localization problem at indoors. Instead, from the multisensor perspective, we thought that a better contribution would be to propose a solution combining already existing alternatives that analyze the measurements from different sensors, taking advantage of their different strengths to solve some of the issues commonly present in the development of indoor localization services.

In consequence, we established the following three main goals:

- **Provide a holistic solution**. Most of the existing approaches had been defined to cover a specific necessity. In contrast, our first goal was to provide a localization solution able to be adapted to different accuracy requirements, device's capacities and environmental circumstances. Our location service had to support a wide variety of applications, from those requiring coarse-grained estimations (e.g. to locate someone with room level accuracy), to those requiring high accuracy estimations and rapid response time (e.g. augmented reality AR applications that work in real time and need centimeter level precisions). Because of their high availability in common smartphones, we focused on the study of sensors like the WiFi interface, the camera and the inertial sensors (digital compasses, accelerometers and gyroscopes). To reach this goal we had to follow a sequence of stages that ensured an appropriate integration of the mentioned sensors, being aware of the quality of the measurements obtained:

    1. The initial stage consists on the analysis of radio signals, specifically WiFi signals, which allows the development of solutions able to obtain coarse-grained estimations with errors around a few meters. Sometimes the accuracy obtained can be considered good enough to fulfill the requirements of some location-aware applications.
    2. We would integrate the use of the images collected by the camera within our localization solution. In previous studies, the analysis of the features extracted from these images had demonstrated it usefulness to accurately estimate the camera pose. Using simple fingerprinting techniques to look for coincidences among a database of features, or using more sophisticated

techniques of image processing, the integration of this information should allow the improvement of the accuracy of our estimations.

3. Our initial proposal to accomplish the multisensor integration consisted of the utilization of WiFi and compass measurements within the process of image analysis. The coarse-grained estimations obtained from WiFi signals analysis, supported by the analysis of compass measurements, could be a good filter to focus the image analysis in the tentative zone of the entire scenario. As it is well known in the computer vision field, the process to look for coincidences between a source image and a huge set of referenced images is computationally intensive. Therefore, reducing the number of images to compare with is an important challenge that must be addressed in order to provide rapid response time, supporting applications that require location estimations in real time.
4. Finally, our multisensor idea included the use of inertial sensors to support the localization process during the on-line phase. Existing solutions based on the analysis of inertial sensor measurements allow to track a device during a short period of time with relatively good accuracy. A reference position must be previously established using other sensor information for an appropriate functioning.

- **Improve the training phase**. One of the most important drawbacks of localization systems is the time and the human resources required to carry out the training phase in order to build the sensor models. Different proposals try to solve this issue providing techniques for the automatic creation of fingerprinting maps based on the analysis of inertial sensor measurements using dead reckoning techniques that infer the path covered by the operator. However, when we set out this problem none of them considered the use of the camera, which implied an important challenge because of the restrictions to ensure the capture of images in optimal conditions, and the added difficulty of the complex image processing required.

  Consequently, our second goal was to design a solution to improve the system training with regards to the interaction demands of the operators. This solution had to facilitate a correct capture of the sensor information required to build the fingerprint maps of WiFi signals and images, being able to automatically geo-tag the measurements with the position in which they were collected.

- **Offer a modular and extensible system architecture.** Previous goals cannot be achieved without the support of an architecture that organizes the functionality that must be implemented and the flow of information among the different entities that made up the full system. Therefore, another important goal in the early stages of this thesis was to provide an architecture that supports the multimodality offered by the smartphones. The fact of integrating different sensors within the same solution implied several challenges:
  - We had to be able to manage sensor measurements of different nature.
  - We needed to implement different modules in order to analyze these measurements, ensuring appropriate processing taking into consideration the required computational resources.
  - It was essential that the architecture adequately supported the fusion of all the available information in order to carry out the location estimations in the most appropriate way.

1.3 Structure of this document

In this introduction, we have put the indoor localization into context and have indicated our main goals for this work. In **Chapter 2** we will perform a thorough revision of the related works that we analyzed in the early stages of this thesis and during its development. This study allowed us to identify the pros and cons of the analyzed techniques, realizing that there is still need for more research in this field.

Our work continues then by accomplishing a wide experimental study of different positioning techniques that make use of radio signals, which is described in **Chapter 3**. There we will present a solution which integrates different existing techniques with the main aim of building an alternative that achieves a good trade-off between performance and accuracy.

In **Chapter 4** we will introduce the use of images under the premise that they will allow us to improve the reliability and the accuracy of our estimations. In this initial version of our multisensor approach we will show how radio signal based estimations can be used to reduce the time required to accomplish the image analysis. We will use

the images for scene recognition obtaining high reliable coarse-grained estimations using fingerprint-based techniques. Additionally, we will also accomplish precise estimations reducing the error down to a few centimeters. This high accuracy requires the use of advanced techniques in visual computing, including camera resection techniques which depend on the availability of 3D maps of the scenes.

The generation of these 3D models is a delicate task which can be negatively influenced by different factors, like the presence of similar objects which are distant in the space, or the size of the environment to model. In **Chapter 5** we will provide a solution making use of inertial sensors to facilitate the training phase and at the same time improve the reliability of the 3D model building process and therefore the precision of the obtained models. We will demonstrate that more accurate estimations can be obtained using the models generated using the proposed method.

Once all of our technical proposals have been presented, **Chapter 6** will describe the architecture that organizes all the functionality described in previous chapters. This architecture will demonstrate the modularity of our solution and the capability of adaptation to the different configurations that may be adopted when developing a location aware system.

Finally, in **Chapter 7** we will present the main conclusions reached, defining some interesting research lines that may be addressed to improve this work in the future.

Chapter 2

Related Work

In this chapter we are going to summarize some of the proposals made during the last few years regarding the development of location services and those earlier proposals which have had an important influence over the time. In Chapter 1 we motivated the problem addressed throughout this thesis, and as mentioned, our main aim is to provide localization services for mobile devices moving around indoor environments. Therefore, we concentrate our analysis on those techniques which are somehow applicable within this context.

Moreover, we focus on the use of smartphones and tablets because of the high variety of sensors they integrate, which makes the utilization of a wide range of positioning techniques feasible. This high availability of sensors enables us to provide different levels of accuracy in the position estimations, from coarser-grained estimations which support those applications that require an accuracy of a few meters, to fine-grained estimations in order to serve augmented reality applications which require estimation errors of a few centimeters. Therefore, the techniques analyzed in this chapter are restricted to those that make use of sensors available in smartphones. Specifically we give special attention to some of the most common sensors, like radio, camera and inertial, which will be explored in following chapters.

We will analyze the strengths and weaknesses of different single-sensor and multi-sensors proposals that made use of the mentioned sensors (among others). For that reason, we will carry out an initial revision of single-sensor proposals in order to facilitate a clear perspective of their main characteristics, context of use and achievements. As we will

see, although there are some techniques able to provide a high level of accuracy, most of them lack aspects that make them not suitable for specific situations. Consequently, multi-sensor based proposals which are also analyzed, exploit the positive aspects of each individual sensor, combining them in a adequate way to obtain the maximum benefit in terms of accuracy and scalability.

2.1 Single-sensor technologies used for localization

In this first section we are going to carry out a review about some of the most important single-sensor alternatives that were proposed in order to implement a location estimation algorithm for indoor environments. Different types of techniques have been used during the last years to design efficient algorithms, nevertheless we are going to pay more attention to those based on the analysis of radio signals, images and measurements from inertial sensors, since they can be easily collected using commodity smartphones and tablets in indoor environments.

2.1.1 Radio based proposals

Several methods have been used to infer locations based on the analysis of radio signals. Depending on the problem, each of these solutions has specific requirements as to what type of measurements are needed and the parameters extracted from radio signals which are used. The most common radio technology used is 802.11, and most of the mentioned techniques will be based on this technology. Nevertheless, different alternatives have also been proposed based on other radio technologies like RFID, UWB, or GSM, among others.

2.1.1.1 Lateration (time-based methods)

Lateration methods estimate the position of an object by measuring its distances from multiple reference points. In most cases, these algorithms make use of the signal propagation speed in order to collect distance-related measurements. That is the case of the *time-of-arrival* (TOA) [43, 73, 183] and the *time-difference-of-arrival* (TDOA) [48, 187, 188]

methods, which estimate the positions from distance-related measurements to fixed sensors (emitters or access points) with known positions.

In the first case, the distance from the mobile target to the measuring unit is directly proportional to the propagation time. Nevertheless, in the TDOA implementation the idea is to estimate the relative position of the mobile device by examining the difference in time at which the signal arrives at multiple measuring units (i.e. APs in a WLAN WiFi network), rather than the absolute arrival time of TOA. Despite the good accuracy that can be obtained with estimations of less than one meter of error, these time-based methods require special sensors or hardware to be installed in the covered area due to the requirement of precise synchronization. Though this drawback is alleviated in TDOA, the accuracy is still limited because of inherent multipath fading and *non-line-of-sight* (NLOS) propagation factors which appear in indoor environments.

2.1.1.2 Angulation

Angle-of-arrival (AOA) methods work by observing the angle of arrival of signals. The location of the mobile device can be estimated by the intersection of several pairs of angle direction lines [51], each made up by the circular radius from a base station to the mobile device. In this case 2D or 3D positions can be obtained using two measurements from two different base stations, nevertheless sometimes the case of 3D estimations is solved using three measurements from different APs, over-determining the problem. In this case the synchronization between the different measurements is not necessary. These solutions are able to accurately calculate the device pose with estimation errors around one meter.

Huang et al. [48] propose a hybrid approach making use of the previously introduced TDOA method and the AOA approach. The distinctive feature is the integration of the estimation of the *azimuth AOA* (A-AOA) and the *elevation AOA* (E-AOA). These compound solutions are able to improve the positioning accuracy obtained by either AOA and TDOA separately. To obtain accurate estimations, the collection of precise measurements is required. For that purpose, an alternative is the utilization of directional antennas or antenna arrays as Wang et al. proposed in [92]. Nevertheless, despite the high accuracy it notably increases the cost of the system.

As in the previous case, AOA methods work well in *line-of-sight* (LOS) situations but are susceptible to multipath interference, thus accuracy and precision decrease when confronted with signal reflections from surrounding objects typically present at indoor scenarios.

2.1.1.3 Proposals based on signal properties

Within this category we can find different proposals based on the analysis of the properties that are intrinsic to the signals, such as the signal phase, the *signal-to-noise ratio* (SNR), the *received signal strength* (RSS) or the variation that represents this value in device-specific units as the *received signal strength indicator* (RSSI). In general, most of the alternatives are based on the analysis of the RSSI, and can be classified within two general categories: *radio-frequency* (RF) *signal attenuation* methods, and *fingerprinting* methods.

RF signal attenuation methods

RF signal attenuation-based methods utilize signal theory to define propagation models that estimate the distance between a wireless device and the observed access points. This type of methods is frequently classified as an special type of lateration approach, which can also be performed by using RSS, based on the knowledge of the transmitter output power, the antenna gain and the appropriate path loss model.

The complexity of signal propagation in indoor environments has been analyzed by Chey et al. [45], Zhao et al. [196] and Phaiboon [143]. The main difficulties derive from the complexity of extracting the attenuation factor that influences the RSS transmissions. In [157] Seidel and Rappaport derived the *floor attenuation factor* (FAF) model to estimate the distance between the mobile device and the APs in multistory buildings. In the RADAR system [23, 24] authors disregarded the effects of the floor and presented the *wall attenuation factor* (WAF) which determines the distance to an AP according to the attenuation caused by the walls obtaining estimation errors down to 5 meters. One of the disadvantages of these proposals based on propagation models is that in most cases there are some parameters employed which are site-specific and that change dynamically. Thus they cannot be directly applied to different environments.

As in the case of lateration methods based on transmission time, the localization accuracy of RSS attenuation based methods are significantly affected by the unpredictable setup in indoor environments. The signal propagation between the wireless device and the access points suffers from the connectivity of LOS, *non-line-of-sight* (NLOS), and the shadow fading due to the complications of indoor environments, such as walls, furniture and the movement of people around.

Fingerprinting

These methods are based on radio maps containing patterns of RSSIs which are obtained using 802.11, Zigbee, Bluetooth or any other widespread wireless technology. These methods can be considered as empirical approaches, since they do not make any assumption about the environment or signal propagation paths. Systems based on fingerprinting work in two phases. Firstly, an *off-line or training* phase is performed to build a labeled map of radio signals of the environments. Then, it is during the *on-line* phase when the system is ready to estimate the user location. One of the main difficulties encountered by this method is the definition and maintenance of the fingerprinting maps that guides the localization process. These maps can be manually obtained by collecting signal samples, but it is not an efficient process in large scenarios. Nevertheless, the maps can be derived from radio propagation models as indicated by Hashemi in [78] and Rappaport in [149], or can be obtained automatically by using crowdsourced (organic) approaches like those presented by Park et al. [142], Schmid et al. [155] or Radaelli et al. [146] where users consequently contribute with location-blind measurements while using the localization services. Crowdsourcing proposals have recently emerged as a viable solution to address the definition and maintenance of fingerprinting maps. However some early systems employed the same idea to expand a minimum radiomap created during the training phase. The ActiveCampus [72] project was a clear example of this type of solutions.

Other solutions to address the map creation are the *simultaneous localization and mapping* (WiFi-SLAM) approaches. Fox et al. [59] proposed a method which uses the *gaussian process latent variable model* (GP-LVM) to relate RSS fingerprints and models human movements (displacement, direction, etc.) as hidden variables. When a small

portion of RSS measurements are tagged with the real coordinates, semi-supervised localization estimate the new location of the new measurements according to RSS dissimilarity. GraphSLAM [90] further improves WiFi-SLAM regarding computing efficiency and relying assumptions.

Among the different types of fingerprinting algorithms which have been already proposed we highlight those belonging to these three categories: deterministic methods, probabilistic methods and approaches based on neural networks. From now on, we are going to focus on 802.11-based solutions.

Deterministic techniques represent the signal by a vector of scalar values (according to the number of APs considered) and use some pattern-matching method to estimate the user location, as for example the k-nearest-neighbor algorithm. One of the most widely known alternatives is *RADAR* [23, 24], whose authors proposed an empirical method that estimates the position of a mobile device making use of a fingerprinting database. They studied the influence of the number of measurements used to calculate the position and orientation of the device over the final accuracy obtained. An alternative to this classical approach is presented by Asim et al. in [161, 162], which proposes a localization method based on the analysis of the variability of the signal intensity over time. Though relatively simple in implementation, these methods do not provide really accurate estimations (around 5 meters of estimation error in real scenarios) because they obviate some of the common drawbacks related to the signals transmission at indoor environments, such as the multipath fading which causes interferences and affects the signal reception.

Probabilistic techniques store information about the signal strength distributions from the access points and represent user positions as probability vectors. In [114–116] Ladd et al. proposed a grid-based bayesian system and integrated a sensor fusion step based on a *hidden markov model* (HMM). This work was extended by Haeberlen et al. in [75] where the authors presented an in-depth comparison of the accuracy obtained making use of the bayesian method with a histogram-based sensor model and a parametric-based sensor model. Additionally, different alternatives were presented to calibrate the mobile devices, improving the system compatibility with heterogeneous devices with different signal reception capabilities. Finally, within this category we can also mention the Horus system [192–194] as a refinement of previous approaches. Horus

employs a stochastic description of the RSSI map and uses a maximum likelihood based approach, introducing the usage of clustering techniques to minimize the amount of operations to be performed to accomplish a position estimation. Because of their a priori good behavior, as we will discuss in Chapter 3, we have tested some of these probabilistic techniques, integrating them into a global solution which provides good results in terms of performance and obtaining accuracies down to 3 meters of estimation error in real time tests.

Neural Networks methods differ from the previous ones in the way the model is generated during the off-line phase. These techniques make use of the measurements that made up the fingerprinting map to train neural networks. During the on-line phase, the real-time collected RSSIs are used to build the input vector for the neural network system. Castro and Favela [39] proposed the use of Recurrent Neural Networks (RNN) [129], specifically the Elman networks alternative, to map the signal strength to 2D coordinates. Most common implementations are based on the use of a Multy-Layer Percentron (MLP) [27, 153]. As a flexible model, the MLP neural network requires no a priori knowledge of any environment parameters such as the location of access points and building characteristics. The accuracy, in the range of a few meters of error, and the performance of these methods were reported to be better, although not significantly, than the deterministic solution. However, the disadvantage of neural networks lies in their slow training time which may require large training sets to get relatively accurate location estimations.

Compared to other types of positioning techniques based on other sensors, fingerprinting is not able to provide the centimeter accuracy reached by other proposals. However this level of accuracy is not necessarily required for most location-based applications. We will demonstrate the low cost, in terms of computational resources and time, which implies the estimation of the position using these techniques, what is an important starting point for the integration of other sensors that will be useful to refine the location.

Managing device heterogeneity in fingerprinting localization systems

One of the imposed requirements of every location-based system is the support of heterogeneous hardware clients. The wide variety of devices, the lack of standardization and the diversity of hardware they integrate make them behave in a different way depending

on the reception of radio signals. The gain of the antenna, the reception power and the firmware used, provoke notable differences in the values of the RSSIs observed. These differences negatively affect to the accuracy obtained by those methods based on the analysis of the RSSIs. Consequently, it becomes necessary to include some mechanisms to address this problem, avoiding the use of specific hardware for an appropriate functioning of these systems. In the literature we can find different proposals which try to solve this problem in different ways.

Homogeneous radiomaps can be built from RSS data collected with a single device or by several contributors carrying the same devices. Alternatively, Cheng et al. [42] present a clustering algorithm to group similar devices in clusters, so that they can share the fingerprints among them. In case a new device wants to contribute data to the system, they employ an *expectation maximization* algorithm to learn the linear fitting parameters for matching the best cluster.

In the same way, Haeberlen et al. [75] proposed different solutions to address the calibration of the devices in different ways (manually and automatically). Despite the higher cost of the manual calibration processes, they provide better accuracy results than performing automatic calibrations, obtaining estimation errors down to 5 meters using probabilistic approaches for localization.

Another solution is the one presented by Kjærgaard [101] where the author proposed a manual method and an automatic learning-based method to find the calibration parameters of a specific device. In [102] the same author addresses the heterogeneity of devices by considering signal-strength ratios between pairs of base stations instead of absolute signal strengths, demonstrating the benefits of these techniques obtaining similar results to those using a manual calibration.

A more recent alternative is the FreeLoc system [189], which handles heterogeneous data by using relative, rather than absolute, RSS values in the radiomap fingerprints. Additionally Lyu et al. [41] use the RSS order and the dependency between the measurements obtained by different devices as two different features to support the estimation of heterogeneous devices. However, in these type of solutions the fine-grain information of the RSS values is lost and the quality of the radiomap may deteriorate. In Chapter 3 we address the problem of the heterogeneity devices indicating the specific solution we have used within our system which is mainly based on performing a manual calibration.

2.1.1.4 Proximity

Proximity-based techniques are commonly used when there are no important requirements regarding the desired accuracy. Nowadays there are a huge number of applications which only require proximity information and not absolute location, as for example those used to discover nearby resources or to detect surroundings users to build an ad-hoc social network. Rather than providing fine-grained estimations, these systems provide only coarse locations, perhaps the nearest room or a section within a building.

In [181, 182] Want et al. proposed the *Active Badge* solution. People were tracked by wearing small computing devices (badges), each of them associated to a globally unique code that was periodically broadcasted through an infrared interface. Infrared (IR) signals were picked up by a network of sensors placed around the building. By determining which badges were seen by which sensors it was possible to deduce the location of a badge with a room-level accuracy.

Bluetooth-based commercial solutions like the *MvixAir Bluetooth Marketing System* [6] was designed to distribute advertising content to people who own devices that use Bluetooth wireless technology that are within relative proximity of retail stores, for example inside a mall. Other proposal that makes use of Bluetooth was made by Banerjee [25]. The main drawback of using a Bluetooth-based solution is that this technology suffers serious transmission problems in crowded areas due to signal attenuation and interferences.

Another solution based on technologies with fewer coverage restrictions is the *Nearme* approach [111]. *NearMe* is a service allowing pairs of devices to compare the 802.11 radio signatures they hear to decide whether they are in physical proximity to one another. This solution has been used to implement applications such as finding friends that may be in the surroundings, closed devices like printers, requiring estimation errors around 10 meters. Another proximity solution based on 802.11 radio signals is the one proposed by Kjærgaard et al. in [103]. This solution is able to infer devices proximity and separation with an estimation error of a few meters in more than 94% of tests done.

Proximity solutions have been also deployed for outdoor environments making use of other technologies with higher ranges of coverage, as in the case of GSM. Trevisani and Vialetti [174] evaluated the quality of localizations based on the identification of the

GSM cell that the mobile phone is connected to. As expected, the obtained accuracy was extremely variable, achieving estimation errors of hundreds of meters in some cases. However, the main advantages of this approach are the full coverage which it provides, and the fact of not requiring modification to either the cellular handset or infrastructure.

An example that includes the use of WiFi and GSM systems is Placelab [117], which relies on proximity detection of permanently mounted tags to locate mobile readers. Though mainly meant to work outdoors to support the lack of GPS signals reception in canyon environments, this solution has also been tested in indoor scenarios, providing high availability and achieving accuracies of around 15 to 20 meters.

To summarize, the advantages of using proximity systems are that, unlike other location sensing techniques, they do not require any a priori geometric calibration of the environment where the system is going to be used, they are not computationally expensive and they can be deployed without requiring high investment costs. Nonetheless, accuracy is not one of their strengths, and depending on the technology selected they need a high density of readers to gain reliable ubiquitous coverage, which obviously also affects the obtained accuracy.

2.1.2 Vision based proposals

Within the localization research field, there are a large amount of proposals which focus on the utilization of the contextual information captured by cameras. The ubiquity of cameras, which have been integrated in smartphones and other wearable devices, allows us to consider this sensor as an important source of information to develop systems that provide a better accuracy than the ones shown in previous sections. These proposals can be classified according to different criteria. A first classification can be done depending on the final goal, which divides these techniques in three different categories: *Person-Detection* [26, 94, 122, 160, 163] which are typically deployed for security scenarios and rely on background information subtraction; *Tracking-Algorithms* [53, 154, 191] which locate human presence and position in indoor environments using multiple camera systems; and *Object-Recognition* [126, 173, 179] whose main aim is to identify something or someone among a set of reference images.

An alternative classification can be made based on the interaction between the cameras and the environment. We distinguish two different categories: *Landmark-based* solutions [58, 79, 134, 138] and *Landmark-free* solutions [18, 93, 110, 185]. In the first case, the proposed solutions require the existence of dedicated position-annotated coded markers which are spread around the environment to support the localization process. The use of these landmarks implies the modification of the normal aspect of the scene, what is an important issue in some environments. Moreover, the accuracy obtained will depend on the number and density of the landmarks deployed.

On the contrary, in the landmark-free solutions there are no imposed restrictions regarding the necessity of deployment of such landmarks. They rely on sequences of reference images taken from the scene. These images are analyzed to extract features that provide distinctive information of that scene. In general, these features are employed to build the search structures that will be used during the localization stage. Then the real-time positions are estimated based on the utilization of the features obtained from real-time images and the execution of matching algorithms to compare them with the features obtained from reference images. The main drawback of these techniques comes from the tedious process of image collection to train the system. However, these types of systems are able to reach high level of accuracy.

After analyzing the pros and cons of each category, and according to the requirements of our system, we decided to focus on the study of those proposals based on the extraction of image features for object recognition in landmark-free environments.

2.1.2.1 Image features extraction

We analyzed different alternatives to solve the detection of distinctive features from images. *Global features* extraction methods, like the one proposed by Swain and Ballard [168], were based on the identification of distinctive colors in the full image composition. Though being able to correctly identify scenes with a hit probability rate close to 100% in favorable conditions, the fact of obtaining a global description of the entire scene can be severely affected by image clutter and occlusions. These drawbacks limit the use of this technique to cases with clean backgrounds or where the object can be segmented out.

Image segments identification proposed by Carson et al. in [38] minimizes some of the issues detected by global features. These methods are based on the detection of regions or segments in the image that correspond with a single object. They are based on characteristics of color and textures of the objects that appear in the image. One factor impeding the utility of segmentation for recognition is the unsatisfactory quality of image segmentation algorithms.

Exhaustive sampled features approaches like the one made by Carbonetto in [37] is another alternative that can be found in the literature. This method has been proposed with the main aim of solving the problems encountered with global features or image segmentation. It is based on the exhaustive sampling of different subparts of the image at each location and scale. The main drawback of using this approach is the complexity to find the most appropriate grid of patches to carry out the sampling process. Moreover, the limitations to achieve invariance to geometric object deformation is one of the causes that make this method not suitable for those situations in which camera movements are not restricted, as happens in our context.

During the last years the techniques based on *local invariant features* extraction have gained importance, and have been integrated in a wide variety of solutions. Among their advantages we highlight the capability to find correspondences in spite of large changes in viewing conditions, occlusions, and image clutter. The good performance of these techniques has made them the most utilized within the field of the image recognition. Among the different proposals made for the extraction of local invariant features, one of the most important is the *scale invariant feature transform* (SIFT) proposed by Lowe [126, 127], an image processing technique suitable for object recognition. It provides a powerful mechanism to perform robust matching between different images of a given scene. One of the most important characteristics of SIFT is that each feature is represented by a 128-dimensional vector, making them highly distinctive, as it allows a single feature to be correctly matched with a high probability against a large database of features. Different algorithms and refinements of the original version have been proposed by Vedaldi et al. [177] and Turcot et al. [176]. In Chapter 4 we will give a detailed description of this technique, since it will be integrated within our system. We will also compare and evaluate different matching techniques [20, 31, 120, 137] that can be used to carry out the comparison.

Among the large amount of proposals that we can find in the literature, we have evaluated other alternatives as *affine-SIFT* (ASIFT) proposed by Goushen and Morel in [195], *Speeded Up Robust Features* (SURF) introduced by Murillo et al. in [139] and Bay et al. in [28] or *Features from Accelerated Segment Test* (FAST) approach suggested by Rosten et al. [151]. Let us make a brief description of each one to assess their advantages and disadvantages.

ASIFT tries to solve the problems of SIFT in order to support changes on illumination and viewpoint. It simulates viewpoint changes in order to reach affine invariance, by varying the camera orientation parameters. However, as it was demonstrated by Xun et al. in [123], ASIFT obtains less tentative matches compared with SIFT when dealing with images with small angular changes. Thus, in normal conditions SIFT is the most appropriate option, since the number of obtained matches influences the positioning accuracy. Nevertheless, ASIFT can be used as an alternative option in those situations where SIFT fails to get a good result because of large viewpoint changes.

SURF is another technique inspired by the SIFT descriptor which improves the speed in every step of the feature extraction algorithm. Experimental analysis carried out in [28, 139] demonstrated that this technique is 3 times faster than SIFT while its performance is comparable to SIFT.

With difference to the previous solutions the FAST algorithm is based on the detection of corners at images. As it was mentioned by Jeong and Moon in [95] FAST can be a good solution to carry out the features extraction process in devices with reduced computational resources, as for example smartphones. However, the quality of the features obtained is still far from the one that can be obtained making use of other alternatives such as SIFT.

To summarize, independently of the selected technique the goal is essentially always the same: to extract a collection of visual features from input images, which are reasonably invariant to changes such as translation, scaling, rotation, and illumination. The features are used to perform matching against a database of features extracted from a large collection of images previously obtained from the application environment. Since our main aim was to demonstrate the usefulness of the object recognition techniques to support the localization process, we selected the SIFT technique to be integrated within our multisensor system. This choice is justified by the good accuracy provided and the

reliable implementations that are currently available, which ensured a fast integration within our system. Nevertheless, as we will mention in future chapters, our proposal is not restricted to this specific type of features, and it could be easily adapted to use any other alternative.

2.1.2.2 Visual localization based on 3D reconstructions

Until now, all the techniques and systems mentioned make use of image features to accomplish location recognition based on the use of image matching algorithms. Despite their usefulness, the main drawback of using these techniques is that the accuracy obtained is dependent on the granularity of the fingerprinting map built making use of the reference images.

Going a step beyond, an alternative is the use of three dimensional (3D) models to represent the environment in a more accurate way. Vision-based 3D modeling techniques have gained popularity during the last years because of the possibilities they offer, for example, providing accurate positions for augmented reality applications.

Within this field we can find different approaches. On the one hand, RGB Depth (RGB-D) cameras can be used to build dense 3D maps of indoor environments as proposed by Henry et al. in [52, 81, 82]. The results obtained using these methods are high quality reconstructions. The main drawback we found is the specific hardware required, such as depth cameras (Kinect style) to scan the scene. Moreover, it has some limitations regarding the depth of the scene, no more than 6 meters, what is an major drawback when building indoor scenarios with large open areas.

On the other hand, images obtained from conventional cameras can be used to build accurate 3D models of the scene making use of specific computer vision techniques. *Structure from Motion* (SfM) [77] reconstruction approaches enable the creation of large scale 3D models of scenes making use of the information extracted (features descriptors) from reference images. Such models are more expensive to generate but they help the system to be deployed globally and have been successfully used for localization as it was shown by Li et al. in [124] and Furlan et al. in [64]. The main issue is that 3D models require more storage and processing capabilities than previous solutions. Nevertheless, taking advantage of the high computation power provided by commonly used *Graphic*

Processing Units (GPUs), nowadays it is possible to build systems able to execute the reconstruction process in a few hours. That is the case of VisualSFM [11], an application for 3D reconstruction using structure from motion techniques, that we will use to build the 3D models we mention in Chapter 4.

Once these 3D scene representations have been obtained, they can be used for accurate image-based localization making use of camera resection techniques [60, 61, 77], which is the process of estimating the full six degrees of freedom (pose and rotation) of a camera that produced a given image captured within the modeled scene. In Chapter 4 we will evaluate different techniques and we demonstrate that the computation of this 3D model of the environment will allow us to provide a much more accurate estimation.

2.1.3 Inertial sensors based proposals

Tablets and smartphones are equipped with inertial sensors such as magnetometer (used as compass), accelerometer or gyroscope. These *inertial measurement units* (IMUs) provide estimations of direction, acceleration and rotational velocity, respectively. The work on *indoor navigation systems* (INSs) based on inertial sensors owes much to the robotics community, but it is significantly more challenging in our context since the robot is replaced by a user, whose movements are more complex to control. Now we analyze different alternatives that have been proposed to integrate the use of inertial sensors within the localization field.

2.1.3.1 Dead reckoning

There are some works that track a device from a known initial position using *dead-reckoning* (DR). In DR-based solutions inertial sensors measurements are integrated over the time to infer the movements performed by a user carrying the required hardware. The premise of DR is that given the localization of a person at time t, then the position of that person at $t + t_\epsilon$ can be estimated by simply integrating their known velocity, or what is the same, by twice-integrating their acceleration, during the time interval from t to $t + t_\epsilon$.

Most of these algorithms, like Beauregard and Hass [29] Stirling et al. [166] or Krach and Robertson [108], make use of the accelerometer magnitude and compass measurements

from foot mounted sensors to estimate the stride length and the heading of each step. Using measurements from accelerometers, but this time considering those integrated in smartphones, Pratama et al. [144] proposed a method for step detection that approximates the results obtained using more precise sensors. Angular velocity thresholds have also been proposed by Cavallo et al [40] where gyro measurements were used to infer the followed path in outdoor environments. In this solution, when GPS position fixes are available sensor biases can be estimated and a coarse initial alignment can be performed. Bong-su et al. proposed a solution which joins both technologies in [46], where the position is calculated using the velocity and orientation that are calculated from accelerometers and gyroscopes measurements mounted in robots. Finally, Sharp and Yu [158] proposed a hybrid DR method that integrates information from the accelerometer, compass and gyroscope available at commodity smartphones. They are able to estimate the stride length and direction of movement up to a few cm of error while covering paths longer than one hundred meters.

The common issue of DR systems is the dependence on an initialization source. These techniques can offer good short-term tracking under certain circumstances, though regular absolute position fixes from complementary systems will be needed to ensure long-term operations. Moreover, measurement errors are inevitably present within the sensor data, even more when using noisy sensors as those integrated in smartphones. These situations provoke drift errors over time caused by the inaccuracy of the measurements. Due to the high cost of the reliable sensors, the current tendency is to propose DR solutions based on not-foot-mounted sensors (such as those integrated in smartphones). However this problem is still unsolved on account of these integration errors.

2.1.3.2 ZUPT integration

When dealing with noisy sensors, even small errors in sensing are magnified by the double integration, causing a quick divergence of the location estimations. To minimize the drift in these systems, it is necessary to regularly close the integration loop by applying external constraints. In order to do it, these techniques take advantage of the periodic stance phase of the foot at each step while walking to correct these errors.

The most useful constraint is provided by *zero-velocity updates* (ZUPTs) solutions. ZUPTs assert that the sensor is stationary and can be applied during the stance phase

indicated by the sensor, which must be attached to the foot to detect that situation. The application of ZUPT means that open loop integrations only occur during the swing phase of the foot to which the sensor is attached. For such short durations, drift accrual is limited and longer tracking durations are thus feasible. For a reliable output, however, ZUPTs must only be applied when the foot (and hence the sensor) is completely static.

Proposals like Colomar et al. [49] and Foxlin et al. [62] are mainly based on the use of foot-mounted IMUs to take advantage of these situations. An intelligent integration scheme, that applies zero velocity updates when detecting a stance phase, together with a carefully designed *extended Kalman filter* [80], is able to accurately estimate the position of a pedestrian for relatively long walks.

The great inconvenience of this approach is that it only works properly with higher quality inertial sensors than those commonly found in typical smartphones and tablets. Furthermore the sensor has to be mounted in the shoe of the operator, in order to take advantage of the ZUPT trick. These requirements clearly limit the applicability of this technique in several cases, for example when the operator must hold the device in their hands to take pictures of the environment, as we will discuss in Chapter 5. There we will propose a technique following the same philosophy, which allows the user to detect when the device is static, even when the user is holding it in his hands.

2.1.3.3 Particle filtering based on environmental restrictions

As we have mentioned, one the drawbacks of DR solutions comes from the necessity of using position fixes to ensure a proper functioning. This drawback has been addressed by integrating particle filtering and map constraints. Davidson et al. [50] and Woodman et al. [186] proposed two different implementations to solve tracking problems inside multi-floor buildings, while Sung and Kim [167] proposed a solution to improve the position accuracy of pedestrian in urban canyon environments where GPS is not precise enough. In all these solutions the position of the operator is represented by a set of weighted particles evolving based on the sensor readings (just as the user walks) and constrained by the topology of the environment.

These particles represent the uncertainty in location, stride length, and orientation for each user in this context at a certain timestamp. Initially the position of the particles

is randomly assigned, which implies that some positions are perhaps more likely than others. Each particle contains a weight value that represents the probability of being correctly positioned based on external information, that is according to its position inside the map and taking into account some constraints like its distance to the walls, forbidden areas, or physical restrictions (i.e. it is not possible to cross a wall). When a new measurement is obtained, particles move to a new state, considering the information from sensors and introducing a random variance component. During long walks, the uncertainty in location is reduced with time using these map constraints as main source of probabilistic inference. Then, once the uncertainty has been sufficiently reduced, a backward belief propagation can be performed to trace back the covered pathway.

The solutions based on this technique still present deficiencies related to the error introduced by noisy sensors. Though it can be alleviated by increasing the variances associated with each step event, this is not a satisfactory solution. In turn, this will require an major number of particles to better represent the underlying probability distribution.

Particle filtering is able to capture multimodal distributions which tend to occur when there is uncertainty on which part of the building the user is. By carrying forward multiple possibilities, the particle filter is able to adapt naturally to this multimodality. On the contrary, one of the drawbacks of using particle filters is that they require greater storage and processing resources than other techniques. The minimum number of particles required at any moment is related to the uncertainty in the initial user position and the environment size. However, there is no way to estimate the optimal number of particles beforehand.

Rai et al. [147] presented a novel proposal able to track users without any previous knowledge about the user's initial location, stride-length or device placement. This SLAM solution was done in order to reduce the training effort required to build WiFi fingerprinting maps by means of crowdsourced data, tracking the operators during the training phase. We consider this work to be a valuable contribution, and some of its proposals will be integrated within our system.

2.2 Multisensor proposals

In the wide review of proposals throughout the previous section, we observe that many types of data and methods have been used to infer location. There are important contributions that achieve good accuracy making use of a single sensor, though better results are obtained by multimodal approaches which integrate the information captured by several sensors. Despite in previous researches the multimodality is not one of their strengths, some of these proposals made use of additional sensors at specific moments to support the localization process.

In this section we review some early sensor fusion examples to illustrate some of the benefits of the multimodality which support future design decisions, with the main aim of detecting their strengths and weaknesses. Most of the proposals herein described refer to pure localization systems which integrate several sensors. Nevertheless we will describe other multimodal alternatives that have been proposed as solutions to improve the context data acquisition, or to accomplish activity recognition, among others. Some of the ideas herein commented have been actually put into practice in order to build an extensible, accurate and scalable system, which solves some of the problems that will be discussed and still remain unsolved.

2.2.1 Localization integrating radio signals and images

Hattori et al. [79] use WiFi signals and image recognition to calculate the position of the devices with an accuracy around 3 meters of estimation error. WiFi based estimations are used to obtain a previous coarse estimation to reduce the space search over the map images. Despite not obtaining highly accurate positions, we consider this solution a valuable contribution to solve the problem related to computational cost derived from the image analysis. Nevertheless, the main drawback we found is that image-based positioning is based on two-dimensional visual landmarks that must be placed in the scenario of interest, which can be considered inappropriate in some cases because it involves the inclusion of obtrusive elements in the environment.

SurroundSense [22] is a multisensor location system for indoor environments via ambient fingerprinting of radio and visual attributes, among others. One of its strengths is the ability to discriminate adjacent places that share similar radio beacons, and its capacity

to group semantically closer places (e.g., stores of the same franchise that look and sound similar). However the optical recognition techniques proposed in SurroundSense are too limited, since it only considers pictures of the floor in order to extract information about light and color. Moreover, we consider this approach not appropriate for obtaining accurate positions in large scenarios with uniform aspect, such as the ones that we are going to deal with.

Arai et al. [18] proposed the *Wi-Foto 2* solution, which performs a preliminary analysis of how the integration of techniques like SIFT can be used to improve the accuracy in landmark-free WiFi-based systems, but still obtaining estimations up to 1 meter of error. A similar approach is MoVIPS [185], which uses SURF for image feature detection but also uses coarse-estimations based on WiFi to reduce the size of the candidate set of images to be analyzed and therefore speed up the system. The accuracy obtained by MoVIPS resulted in a median position error close to 1 meter. These works are far to provide highly accurate estimations to fulfill our requirements to support augmented reality applications.

Other multisensor approach based on the combination of radio and visual information is the one made by Miyaki et al. in [133], which describes an object tracking system for indoor/outdoor environments. In this solution authors proposed to fuse information from CCTV cameras and WiFi signals using a particle filter method to combine these two different kinds of sensory input. This solution is able to estimate the user's position with high accuracy but it is not able to obtain the device pose, which is paramount to our interest. Furthermore, one drawback of this proposal is the need for the integration of the localization system with the surveillance system, which in some cases could be considered as a intrusion on the security of the companies. Finally, this kind of solution has to take into consideration the privacy concerns related to the recording of people moving around the scene.

2.2.2 Localization integrating radio signals and inertials

One of the most widely known approaches that fuses these two types of sensors is COMPASS [100], which takes advantage of WiFi infrastructures and digital compasses to provide low cost and relatively accurate positioning services, down to 2 meters of estimation error. The major contribution of the COMPASS system is the study that

demonstrates the influence of the user orientation within the location sensing process. The user orientation is measured by a digital compass to reduce the human body blocking influence in the positioning process.

Other additional multisensor proposals try to compensate for the inherent limitations of inertial tracking discussed in Section 2.1.3. That is the case with the solutions proposed by Miller et al. [131] and Evennou et al. [55] where inertial DR solutions with RFID and WLAN sensor's measurements are combined, respectively. As we discussed previously, inertial tracking systems inherently drift over time and produce errors in position, especially where inexpensive and lightweight systems are concerned, as those integrated in smartphones. Corrections in the position solution at certain points along the path can limit the maximum error to an acceptable level. The main goal behind the integration of radio technologies is to minimize these errors, since known and fixed positions can be obtained using radio-based estimations, for example by the detection of an infrastructure RFID device. Moreover the integration of several sensors improves the performance of the whole system in terms of positioning, and it allows the device to be tracked using DR navigation when radio-based estimations are unavailable. These solutions are able to provide locations with 1.5 meters of estimation error on average while covering paths longer than 100 meters. The limitation of these fusion schemes is that they still rely on both prior knowledge of the environment (maps) and pre-deployment of beacons at known positions.

One interesting integration of these types of sensors is the one proposed by Amendolare et al. [16]. The goal of this system is to provide a robust real-time tracking solution that requires no pre-installed infrastructure and does not need spatial information of the environment. This solution was proposed in order to facilitate the deployment of a localization system for emergency situations, such as a fire in a building. In this solution an *ultra wide band* (UWB) transmitter is sensed by the receiving stations fixed upon emergency vehicles. The receiving stations form an ad-hoc network and establish a local coordinate system. The location system is improved by adding information from inertial measurement units, which allows a better estimation of users positions, obtaining accuracies down to 0.5 meters on average in some of the tested scenarios. Despite its good performance, the main drawbacks we found to be integrated in our solution are the lack of UWB hardware in smartphones and the necessity of having to use highly accurate inertial sensors.

2.2.3 Localization integrating images and inertials

Kourogi and Kurata [107] combine inertial sensors and cameras under the premise that the dead-reckoning deviations can be corrected whenever the camera recognizes the surrounding environment and provides an absolute position estimation. According to the test that the authors carried out, the proposed solutions keeps the estimation error down to 2 meters. Despite of the fact that the obtained accuracy does not adjust to our requirements, we found this proposal interesting with regard to its usefulness to improve the performance of the training phase, as using just a small number of images could improve the estimation of the path followed by the operator, counteracting the noise of inertial sensors.

One variation of this type of solutions, but this time integrating measurements from inertial sensors and images from surveillance cameras, has been proposed by Teixeira et al. in [170, 171]. Here the authors try to find the best matching between the inertial-based estimation and the tracking performed by the cameras. The accuracy reached by this solution is down to 1 meter of estimation error. However, the maximum accuracy obtained by these proposals is still far from what we need, namely centimeter-level. Another drawback is the necessity of integration with surveillance systems.

Aufderheide et al. [21] proposed a solution that fuses inertial measurements and image analysis to achieve an improved performance of the estimation of device pose in terms of accuracy. The position estimations are performed using the camera-based estimations to bound the drift error of the inertial pose predictions for long-term sequences. Moreover, estimations obtained using the IMU data are used to limit the search space for image features matching, and to improve this accuracy while tracking users. Though authors do not provide the real accuracy obtained using this solution, we consider this contribution useful to be integrated within the training phase to automate this process.

2.2.4 Multimodality for context acquisition

During the last years we have observed a notable increase in the number of alternatives that take advantage of the multimodality offered by devices such as smartphones in order to accomplish contextual data acquisition. Crowdsourcing (or crowdsensing) solutions

have been proposed in order to facilitate the collection of sensor measurements to accomplish fingerprinting map constructions. The fact of using the measurements obtained from people moving around the scenario of interest makes this process more efficient from the point of view of the time required to be performed and from its scalability.

Several proposals have been done in this research field as for example the one proposed by Zhu et al. [198], which integrates the use of WiFi and Bluetooth. Apart from using Bluetooth measurements to perform room-level estimations, it is also used to populate the WiFi fingerprinting maps with the corresponding WiFi measurements obtained, that can be coarsely labeled. In [17], Aparicio et al. also presented a similar location system which integrates WiFi and Bluetooth to build fingerprinting maps which fuses measurements of both technologies.

Rai et al. [147] proposed Zee, a crowd-sourced proposal that uses the measurements obtained from inertial sensors in order to estimate the path covered by an operator during the training phase. The estimation made during its functioning allowed the automatic labeling of the WiFi measurements collected. As authors demonstrated, the fingerprinting map obtained resulted in an end-to-end median localization error of around 3 meters when it was tested using the Horus [193] system.

Within this category we can include those solutions designed to perform *simultaneous localization and mapping (SLAM)*, as well as those examples that were indicated in Section 2.1.1.3. All of these solutions can be considered multimodal systems since they integrate the use of inertial sensors with others like WiFi to automatically build fingerprinting maps, as in the case of Faragher et al. [57] , or the camera to generate 3D models of complex indoor environments, as Liu et al. [125]. The main reason that made us discard this proposal to automate the generation of the 3D models is that the authors made use of a high accurate IMU that allowed a precise estimation of the camera pose during the training.

2.2.5 Multimodality for activity recognition

In this subsection we are going to do a brief review of some of the different proposals that make use of the multimodality to extract additional information from user's behavior. All these works take into consideration the information related to the location of the

device to infer contextual information, as for example the recognition of human activities. Though out of the scope of this thesis, we want to show these solutions as a derived field within localization research.

Blazquez et al. [34] proposed the *inContexto* system, which uses the information obtained from sensors embedded at smartphones in order to infer physical actions such as walking, running, and still being able of achieving an accuracy of around 97% correct estimations.

Chon and Chan proposed the *LifeMap* system [47], a context provider for location-based services. This system integrates the measurements obtained from accelerometer, digital compass, WiFi and GPS, in order to obtain high-level information related to the users position, as for example the definition of *points of interest* (POI).

Another multimodal alternative is the one made by Kjærgaard et al. in [104] where authors proposed a novelty solution to accomplish the recognition of flocks which can be used to infer behavior patterns of people that move in groups. The proposed technique is based on the fusion of data from WiFi, accelerometer and compass, being able to achieve an accuracy of around 87%.

This is only a brief summary of the full set of proposals that can be found in the literature regarding the development of multimodal systems. Though they are not purely localization solutions, these works make also use of positioning information as the keystone that allows the association of one specific behavior within the context in which it has been observed.

2.3 System structure alternatives

In this section we give a brief review of different alternatives that have been used to define the architecture of the location systems. The distribution of the functionality among the different components that takes part in the system has an important influence on its scalability and extensibility. The design of these components depends upon where the positioning measurements are calculated and where position information is estimated and used. Implications on the privacy of the information managed, like sensor measurements and position estimations, must be considered. We are going to discuss

different works, identifying the main reasons to design our own architecture in order to support the requirements imposed by our solution.

2.3.1 Classification based on the distribution of responsibilities

In the literature, positioning systems are classified into three categories according to the division of responsibilities between clients, infrastructure elements, and servers. These categories are: Client-based systems, Network-based systems and Client-Assisted Systems. In this section we are going to give a brief description of each, highlighting the pros and cons of using them when designing a location system.

Client-based systems

In client based systems the roving receiver reads the appropriate signal measurements and uses them to determine its position. Hence, a self positioning device knows where it is and its applications can use this information to make position-aware decisions. Some of the previously mentioned works follow this philosophy, like the Horus system [193]. Another clear example could be the GPS system, in which each device calculates its own localization using the signals received from GPS satellites. The main advantage is that privacy is not an important issue, since critical information is not sent out from the device. That is, the sensor measurements which can identify the device inside a specific environment are not transmitted. Additionally the positioning information is only known by the device itself, unless the device is willing to share the data. One important advantage of these systems is that the localization capability remains in the absence of wireless coverage or network assistance, since positions are estimated locally. However, there are some situations in which mobile devices do not count sufficiently on computational resources to handle the measurements obtained, as in the case of images where the features extraction implies an intensive computational process.

Network-based systems

Network based systems, or remote localization systems, count on a set of receivers or *sniffers* at one or more locations which are in charge of collecting the measurements generated by the mobile devices. That is the case in which APs collect radio signals

emitted from clients (Krishnakumar et al. [109]), or when the cameras of a CCTV system capture images to track a person (Teixeira et al. [171]). Anyway, the obtained measurements are transmitted to a central site where they are processed to give an estimation of the position of the mobile device. The pros of using this alternative are that it does not add either extra complexity or cost to the mobile device, as it does not require the installation of specific software or hardware, and finally it can be considered as an asset to obtain valuable information from people behavior without any intervention necessary. The consequences of using this strategy are that it offers an inferior accuracy, it requires additional investment in infrastructure hardware which complicates the system deployment and maintenance, and obviously the user privacy is questionable. Mobile devices can be located without being conscious of that fact. Moreover, though accepting to be localized, the information transmitted can be compromised. Different solutions have been proposed to solve these issues, with the purpose of providing basic security mechanisms to control access to sensor data obtained [87], and to ensure the privacy of users when sharing positioning information by means of anonymity (Gedik et al. [66], Meyerowitz et al. [130], and Hoh et al. [86]), obfuscation (Khoshgozaran et al. [98], and Zhong et al. [197]), among other solutions.

Client-assisted systems

Finally we consider the client-assisted alternative. These systems are based on the premise of communication between the mobile device and a remote server, which will be in charge of estimating the locations, as described in [35]. The superior accuracy, availability and coverage are the main advantages of this alternative. Nevertheless, the main disadvantage comes from the need of interoperability between the network and mobiles, which requires the additional development of communication protocols, increasing the transmission load and battery consumption. As in the previous case user's privacy can be still questionable, but a wide variety of solutions that can be applied to secure the transmissions, for example encrypting the data transmitted using protocols like *secure socket layer* (SSL) and *transport layer security* (TLS), and keep personal information safe.

As we will see in the rest of this thesis, this was the strategy selected to develop our system. This decision offered the possibility to balance the computational tasks between

the mobile devices and the core servers because of our requirements when handling images, whose processing was forwarded to the server side in order to take advantage of the better computational resources available.

2.3.2 Alternatives for the modular architecture

As we have observed in previously mentioned works, the deployment of a localization system sometimes requires a complex implementation. Its design must be supported by an architecture that ensures an efficient interconnectivity among the different elements that take part in the localization process. This is even more important when defining a multisensor based system in which more elements take part in the game, in the way we propose in this thesis.

One of the initial proposals that was focused on the design of a multisensor architecture is the *Location Stack* [71, 83]. The authors proposed a model whose APIs were separated from their implementation details and were shown as a hierarchical distribution. Its main contributions were to provide a system architecture able to support current applications, avoiding reliance on particular sensors and allowing hand-off and sensor fusion, and to provide a set of APIs suitable to support the needs of emerging applications. Among its drawbacks we found that it misses some details of the capability of managing different sets of sensors dynamically within the same system, which is important to support several types of applications with different accuracy requirements within the same environment. Furthermore, this architecture does not suggest how to deal with the requirements of providing different levels of granularity and accuracy. Finally, it does not include a detailed *space model* design, defining the geographical information of the scene which is one of the key factors that defines the usefulness of a location-aware system.

The *space model* defines the relations of hierarchy and adjacency between the different areas in which the physical space is divided. The modeling of the physical environment and the representation of locations have an important influence on the overall system functionality, since depending on the decisions taken the system will be able to provide additional services, as for example navigation services. Moreover it will also have a direct implication on the system accuracy and granularity that will finally be provided. Regarding the design of the *space model*, previous works like Jiang et al. [96], Hu et al. [89], Ye et al. [190] and Becker et al. [30], presented different approaches to represent

location information from either a geometric or a symbolic perspective. In these works the authors indicate the advantages of using each one of these representations, but also describe the drawbacks that they imply. In spite of their design differences, the authors' final decisions converge to make use of a hybrid representation of the space model, that is, to make use of a dual symbolic and geometric representation of the environment. We want to mention that, though we have not developed a complete space model, in Chapter 6 we will show some details about the work we have carried out.

Another interesting architecture proposal is *LOC8* [165]. From the point of view of its layered organization, it has several connections with *Location Stack*, but it differs in some key points. Apart from other differences, in *LOC8* authors clearly separated the *space model* from the *sensing model*, treating the data in an independent way. The *sensing model* (or *context model*) decides about the procedure of sensor collection and how the obtained information is managed, making it available to be used in the localization process. *LOC8* defines a *web ontology language* (OWL) to represent the context and the space models, using a symbolic and geometric representation of the defined regions. The drawbacks of this approach mainly come from the complexity of this metadata definition based on the use of OWLs. Moreover, as in the previous proposal, *LOC8* does not support different sensor configurations in the same system, which reduces its adaptability to environmental changes.

More recent studies like Villanueva et al. [128], Beckkelien and Deriaz [32] also addressed the design of localization systems architectures. The former, a multimodal architecture, is technology independent, scalable and auto-configurable, and provides a high level interface in order to offer location services to external systems in a technologically transparent way. This work concerns a global design model for the whole location based system, and focuses on a particular set of technologies. The latter tries to make the way of obtaining information from different location sources uniform within a unique interface. According to the authors this approach is an extensible system that allows for seamless incorporation of new technologies, offering a standard format for geographical positioning and providing an easy-to-use high level interface. This proposal does not provide an architecture to define an autonomous localization system, but the mechanisms to merge the outputs obtained from others previously developed localization engines.

In contrast to these approaches, the architecture described in Chapter 6 provides a complete definition of the components operating within a location system. We justify our design from the necessity of a complete system able to integrate the information provided by different type of sensors, in order to accomplish with the requirements of each specific location-aware application.

2.4 Summary

In this chapter we have presented a wide revision of different proposals within the field of the localization systems. Monomodal and multimodal solutions have been analyzed, as well as different alternatives to accomplish the design of the system architecture.

To delimit the scope of this research we have mainly focused on the study of those sensors that will be integrated in our proposal throughout this thesis. In that sense, this review has allowed us to assess different works that make use of radio signals, images and inertial measurements. During their analysis, we extracted different conclusions regarding the usefulness of each one of them, with the main aim of taking advantage of their strengths within our own system. Furthermore, the different multisensor systems analyzed helped us to think of an additional proposal that solves some of the found shortcomings.

On the one hand, in view of the problem we try to solve and the requirements imposed, in Chapter 3 we are going to do an in-depth analysis of some of the fingerprinting techniques that have been discussed in Section 2.1.1. Due to the inaccuracy of *proximity* solutions, and the difficulties to use *lateration*, *angulation* or *RF signal attenuation models* proposals in indoor environments, we have decided to discard the integration of these alternatives within our proposal. Being conscious of the limited accuracy that can be reached using a fingerprinting solution, we are going to take advantage of these coarse-grained estimations to constrain the search space for the image analysis.

On the other hand, as we will describe in Chapter 4, the images analysis will be performed making use of SIFT, a technique to extract *local invariant features* that can be used to accomplish a robust detection of objects and scenes. The decision of using this technique is justified by the different drawbacks observed in the alternatives described in Section 2.1.2, and the high availability of resources that will allow us to integrate

this technique within our systems. Because of its characteristics, this technique is appropriate to build accurate 3D models of the scene, what will allow us to estimate the position of the device accurately enough to provide AR services. The main problem of our proposal is the necessity to perform a supervised training phase to get reliable reconstructions, which implies high human and time resources. Therefore we have decided to integrate the inertial sensors in this training phase to infer the path covered by the operator, in the way it was done in some of the proposals described in Section 2.1.3. As we will demonstrate in Chapter 5, the accuracy obtained using these proposals (around 3 meters of estimation error) is good enough to accomplish an automatic labeling of the collected measurements, making the automatic reconstruction of the 3D maps of images more reliable.

Finally, in Section 2.3 we have made a brief revision of the different alternatives available to accomplish the design of the structure of a location system. We have discussed the pros and cons of each one, concluding that the client-assisted alternative is the one that fits best our requirements. Moreover, we have introduced different architecture schemes for the representation of the contextual and spatial data. Throughout the rest of the thesis, but specifically in Chapter 6, we will show the details of the architecture defined.

Chapter 3

Localization based on 802.11

3.1 Introduction

Nowadays, widespread user devices such as smartphones, tablets, smart-watches, and in the future also other wearable smart devices based on IEEE 802.11, include the hardware and software necessary to measure the RSSI of the transmitted packets. These capacities together with the already available wireless infrastructures in large building complexes (like university campuses, airports, or hospitals, among others) make the easy deployment of positioning systems based on this technology feasible. Despite the existence of other radio technologies that have also been used for localization purposes, the high-availability of 802.11 networks makes this technology the most widely used alternative to deploy such kind of systems in large indoor environments. Furthermore, regarding economic factors, the utilization of this technology has a lower setup cost due to these existing deployments.

As a first step to achieve the goals proposed at the beginning of this thesis, we suggest an engineering approach of the analysis, implementation and integration of several existing techniques designed for localization based on 802.11 radio signals, specifically the WiFi standard. We performed an in-depth experimental work to test different proposals with the main aim of analyzing their efficiency in a real scenario. We examined their accuracy and performance, as well as other capabilities to improve the system scalability and the support of heterogeneous devices, which is a key point in this type of systems.

We integrated different techniques in order to design a global solution that provides good results in terms of accuracy and performance. Our analysis was mainly focused on location techniques based on fingerprinting and Bayesian inference, since previous studies demonstrated their suitability for localization purposes when using WiFi signals. The outcome solution provides fine-grained estimations, but also coarse-grained estimations with high reliability. The obtained balance between accuracy and performance is especially interesting since it constitutes a solid basis to integrate other sensors, as we will discuss in the following chapters where WiFi becomes the keystone for our multisensor integration.

The rest of this chapter is structured as follows. In Section 3.2 we provide useful information that facilitates the comprehension of the rest of the chapter. Section 3.3 depicts some details about the scenario where the experimental tests were conducted. In Section 3.4 we provide the results obtained after evaluating the integration of the techniques described. In Section 3.5 we introduce the use of a motion model. We describe a proposal that improves the system accuracy supported by contextual information in Section 3.6. In Section 3.7 we discuss how to improve the system scalability. Section 3.8 describes the support for heterogeneous devices and the compatibility between different radio technologies. Finally, Section 3.9 presents our main remarks.

3.2 Overview of WiFi-based positioning systems

We want to start this chapter introducing a baseline overview of the main characteristics of the fingerprinting systems, as well as some aspects related to the WiFi signals that may affect the system performance. Firstly, we emphasize the importance of the network scanning process, which can be categorized in the following two different approaches: *active* scanning or *passive* scanning.

On the one hand, during an *active* scanning a mobile device examines each available frequency channel waiting for either incoming frames generated by other devices due to data transmission or for the so-called *probe delay* timer to expire. In this case one timer ensures that the mobile device is only waiting for a certain period of time for incoming frames. After that, it uses the 802.11 medium access procedure to gain access to the channel and sends a *probe request* frame. If no response frame is received (during the

following 1 to 7 milliseconds) it proceeds to the next channel, considering the current channel as empty. Depending on the network card used, switching the channel requires 5 to 19 milliseconds [148]. On the contrary, if a frame is received, the mobile device processes any subsequent *probe response* frame until a timer elapses. This timer represents the waiting time (usually around 10 milliseconds) to collect potential additional *probe responses* from other access points (APs) in the same channel. The IEEE 802.11 standard does not define default values for these timers, however, in [19, 178] authors empirically studied the values used by several wireless network card manufacturers. In total, considering the averaged values of the mentioned threshold, an active scan of all channels (considering the 13 channels available in Europe) takes around 300 milliseconds to complete.

On the other hand, *passive scanning* was introduced to reduce the workload of mobile devices and hence to save battery power. As indicated by its name, passive scanning does not require any active communication of the mobile device. During the process, a device listens to each channel and waits for a given period of time. If an AP is assigned to a particular channel, the mobile station should receive a so-called *Beacon frame*. Every AP broadcasts *Beacon frame* on a regular basis (usually every 100 millisecond) to maintain the network. By examining the received frame a mobile device is able to recognize neighboring APs and their capabilities. Once again, note that this configurable value is not defined by the IEEE 802.11 standard. In total, a passive scan requires at least 1.3 seconds to be completed. This is nearly five times the time required for an active scan.

Most of the current smartphones, tablets and other wearable devices make use of this scan method, since as commented, the fact of not forcing the device to send *probe request* frames makes this alternative more attractive as the power consumption is reduced. Sometimes, because of power saving concerns, some manufacturers establish longer scan delays, up to 5 or 6 seconds, varying the time of waiting for a beacon frame at each channel, or perform several rounds trough all channels in the same scan. Due to this limitation, during our experiments the monitoring of the WiFi wireless network was performed using a *passive scanning* approach. We realized that the delay introduced to scan the network may imply a major drawback when high update rates are required, nevertheless the integration of other sensors will allow us not to depend on WiFi to obtain the refresh rate desired, as we will see in following chapters.

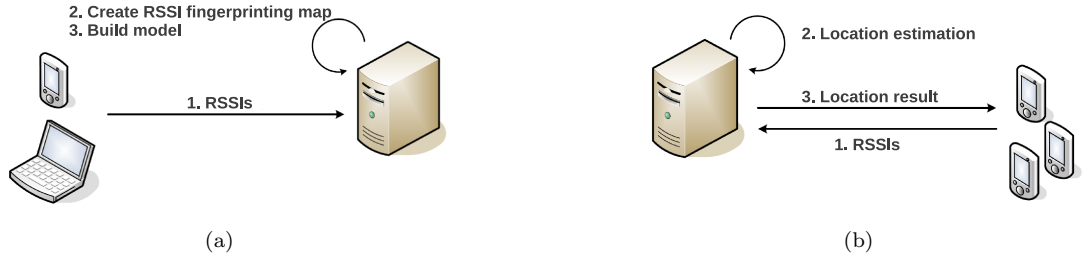


FIGURE 3.1: a) Training phase; b) On-line phase.

After explaining how we scanned the network, we summarize the normal process taking place when deploying a fingerprinting solution. WiFi fingerprint is considered a practical solution since it allows the design of an easily deployable infrastructure-less and low-cost localization system. Every WiFi fingerprinting system works in two phases: an off-line training phase and an on-line localization phase.

An initial *training* phase is conducted in a three-step process, see Figure 3.1(a). Before starting the training process the definition of a grid-layout is required, which partitions the scenario to be modeled in a set of survey positions or cells. Then the first step consists of the collection of multiple WiFi scans with the main aim of characterizing each cell with the location-related WiFi RSSI properties acquired at that point. During the second step these scans are used to build the fingerprinting map of the scenario. Finally, the last step consists of those tasks involved in the generation of the system model.

A fine-grained grid-layout is normally required to achieve high accuracy and resolution. We are aware that this implies significant costs in terms of initial configuration and ongoing maintenance in order to keep it updated to environmental changes and WiFi infrastructure alterations. Depending on the environment, such alterations are not uncommon due to system malfunctions, equipment upgrades, or simply the switching on and off of the WiFi APs controlled by individual users. Although out of the scope of this thesis, different alternatives were proposed in order to minimize these adverse conditions, like those described in [142, 155] based on the incremental updating of the training data utilizing users' feedback as a way of monitoring the changes in the wireless environment.

Referring to the *on-line* phase, Figure 4.3(b), the likely position is calculated based on the current WiFi RSS measurements. Live WiFi measurements will be collected and used

to query for the current position. Using only a few WiFi scans during the positioning phase may generate a large error due to lack of informative RSS data, i.e. the loss of beacon frames from highly distinctive APs, though this issue can be alleviated by using a time sliding window approach. Due to the restrictions imposed by our scanning method, we focused on the use of a single observation in order to prioritize the response time. Thus, the results shown during this chapter can be considered as the minimum value of the system accuracy.

3.3 Experimental environment

As we established during our research, the accuracy provided by the different proposals analyzed is highly dependent on the characteristics of the environment they were deployed in. The propagation of the signals will be very different within an open space scenario (e.g. a mall) from a more complex one (e.g. office building). In that sense, different accuracy results may be obtained applying the same technique.

Therefore, before describing the techniques that we explored and the experiments that we carried out, we consider it interesting to provide a detailed description of the environment where these experiments were accomplished. This will lead to a better understanding and assessment of the work described in the rest of the chapter. The same layout will be applied to order the information shown in the following chapters.

The experiments described in this chapter were conducted in the scenario situated on the third floor of the Faculty of Computer Science of the University of Murcia (see Figure 3.2). The dimension of this environment is 35 meters by 30 meters, and it consist of 26 rooms (offices and labs) and several corridors. We defined a grid layout made up of 94 cells distributed all around the scenario. According to the recommendations made by King et al. in [99] the cells were spaced out 1.5 meters at corridors, defining one or two cells within rooms (depending on their dimensions).

We distributed six different APs along the dependencies, indicated in Figure 3.2 as red crosses. These are six Linksys WRT54G routers with 802.11abg support. The distribution of these APs ensured the coverage of at least four of them in every cell. Since we wanted to evaluate the higher accuracy that can be obtained, we restricted the analysis of RSSIs to those obtained from our own APs. This situation allowed us

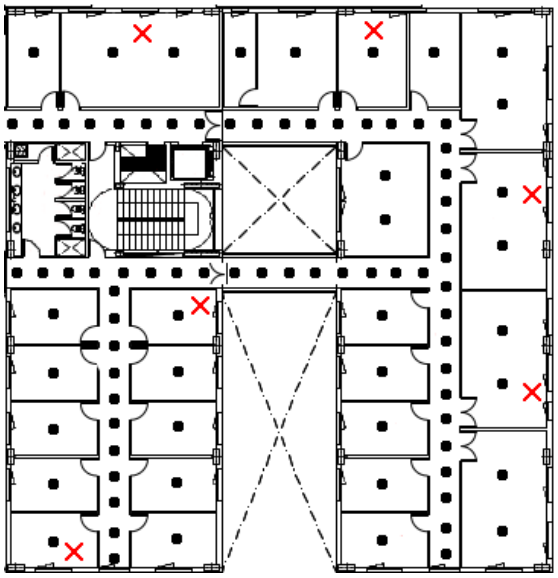


FIGURE 3.2: Experimental environment map. Dots represent the place of each sample point. Red crosses indicate the position of the APs deployed.

to control the status of the wireless infrastructure during the period of time that our experiments took place.

During the training phase we collected around 250 scans at each cell. These measurements were used to build the fingerprinting map to create the system model. Using this large number of measurements was justified as we wanted to perform a complete analysis of the minimum accuracy to be obtained. Nevertheless, in the following chapters we will reduce the training requirements (e.g. the number of WiFi measurements and the level of the layout granularity), since we will use the WiFi sensor just to obtain a coarse estimation of the device position. The training process was completed using an Asus Eee 1201 laptop with a Realtek TRL8191SE Wireless LAN 802.11n card performing passive scans.

The experiments were performed using two different methodologies. On the one hand, to evaluate the mentioned techniques we made use of the RSSI measurements obtained during the training phase applying a 4-fold cross validation method. We partitioned the set of measurements in four different subsets, using three of them to build the system model and the remaining one as the validation set. The results represent the average of the four possible combinations. On the other hand, to test the localization system in real conditions we also carried out additional experiments using different devices: the HTC Legend with Android 2.3 and Samsung Galaxy SII smartphones with Android OS 4.1.2, and one Samsung Tab tablet with Android OS 3.2. We developed the appropriate

software client in order to collect RSSIs and to send them to the server in charge of estimating the device's position. An Intel® Pentium® Dual-Core CPU E2160 was used to host the server.

Finally, in order to test the compatibility with other radio technologies, we deployed a wireless infrastructure based on Zigbee technology, using the TelosB mote platform which implements the IEEE 802.15.4 standard. We distributed six different emitters, placed at the same points as the WiFi APs. For training and localization purposes we made use of an additional TelosB Zigbee mote.

3.4 Analysis of different localization methods

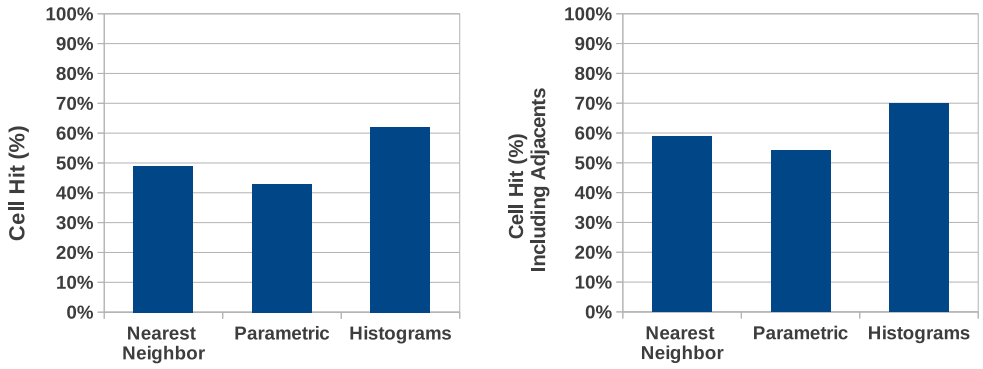
As we previously mentioned, we explored a wide variety of solutions based on 802.11 radio signals already proposed. Throughout this section we describe the progression experimented by our system as we tested different alternatives and integrated them to compose a better solution.

First of all, we performed a detailed analysis of different techniques in order to assess their accuracy and performance. As mentioned in Chapter 2, there is a wide variety of alternatives suitable to provide localization services based on WiFi signals. These alternatives ranged from those proposals based on Neural Networks [15, 56] to those based on Signal Propagation Models [36, 78, 149, 159], including the solutions derived from fingerprinting techniques [24, 75, 194]. Our analysis was focused on the study of the last type of proposals considering those based on deterministic methods, like pattern analysis, and those based on non-deterministic methods, which make use of Bayesian inference. The reasons that led us to take this decision were related to the higher accuracy demonstrated in previous works.

3.4.1 Deterministic methods

Pattern analysis methods have been used to design simple localization solutions. Nearest Neighbor based proposals [24, 136, 164] are simple examples of these techniques. As it was described in previous proposals proved in our own scenario, the accuracy obtained using this method is still limited. We tested this method within our environment in

order to evaluate the suitability of this solution. Results are shown in the first column of Figure 3.3(a). In our experiments we were able to estimate the device localization with a cell hit rate around 50% of tested cases. This value rises up to 59% when adjacent cells were also considered as successful estimations, see Figure 3.3(b). In other words, considering the distance between cells, implies an estimation error around 2 to 3 meters in 59% of cases, under ideal conditions (same device for training and testing and no time variations). In the rest of the cases (around 40%) the error estimation is higher. We are aware that the number of APs considered has a direct impact over the system accuracy when using this kind of method, since a reduced number of APs does not ensure a notable variability of reception in nearby cells. However, in real environments we cannot ensure a higher density of APs allowing a richer variability of reception, thus we consider this technique not suitable to solve our problem.



(a) (b)

FIGURE 3.3: a) Accuracy of different techniques expressed in terms of percentage of cell hits; b) Considering coincidences with adjacent cells.

3.4.2 Non-Deterministic methods

We also tested several non-deterministic solutions in which the position was represented as a probability distribution using a Bayesian inference technique, as it was discussed in [75, 115]. These algorithms estimate posterior distributions and provide better results in those cases where the analyzed sensors have non-Gaussian noise distributions.

We have to take into account that signal propagation in an indoor environment is very noisy since it is affected by reflection, diffraction, and scattering of radio waves caused by structures within the building. These dynamic environmental influences can cause

the observed signal strength to vary considerably, which complicates the estimation of the location using a single and independent signal observation. So, using historical information about the previous locations of the user, we may get better results by means of probabilistic methods.

To perform these estimations we made use of the Bayes' Rule much in the way it was presented in [115], see Equations 3.1 and 3.2. In contrast to the original equation, in our implementation we omitted the first term in Equation 3.1, which considers the frequency of appearance of the APs $Pr(f_\gamma|c_i)$, due to the low number of APs deployed in our testbed and the insignificant differences in the appearance of APs in nearby cells. We verified that it did not show any improvement in terms of estimation accuracy.

We consider $C = \{c_1, .., c_m\}$ as the set of cells that make up the finite space state, where m is the number of cells defined. We also define n as the amount of different APs present in the current observation, π as a probability distribution vector where each element $\pi_i = P(c_i)$ is the probability of being at cell c_i, and $\pi' = P(c_i|O_j)$ as the a posteriori probability of being at that cell given the observation O_j. Therefore, due to the independence of the measurements obtained from the different APs at each observation O_j, the probability to take a measurement from the access point a_β at reference cell c_i with a signal strength λ_β can be expressed by the conditional probability:

$$Pr(O_j|c_i) = \left(\prod_{\gamma=1}^{N} Pr(f_\gamma|c_i)\right)\left(\prod_{\beta=1}^{n} Pr(\lambda_\beta|a_\beta, c_i)\right) \tag{3.1}$$

These conditional probabilities are used to update the probability vector π by applying Bayes' Rule in the following way:

$$P(c_i|O_j) = \pi'_i = \frac{\pi_i Pr(O_j|c_i)}{\sum_{\alpha=1}^{m}(\pi_\alpha Pr(O_j|c_\alpha))} \tag{3.2}$$

We tested two different probability distribution model, parametric and histogram-based models. The parametric distribution was built by modeling the signal intensity as a normal distribution defined at each cell and for every base station by its mean and standard deviation. Histogram-based distributions represent how frequently a signal intensity is observed at each cell and for every base station considering certain ranges or intervals. While the parametric-based models can only summarize the signal intensity

distribution by a best-fit Gaussian curve, the histogram-based models already represent the sensor model explicitly, representing non-Gaussian signal intensity distributions accurately. The main advantage of the first option is the compactness of the model that it generates, but the ability of the histogram to represent some details like the appearance of bias or the multimodality of the distribution makes them more appropriate.

Figure 3.3(a) shows the results obtained within our environment making use of these alternatives. A 10-bins histogram-based distribution was built in order to test the histogram based alternative, where a *bin* represents each one of the disjoint categories in which a histogram is divided into. In our initial tests we divided the signal intensity range of values into 10 different categories.

During our experiments the best results were obtained using the histogram-based solution, see Figure 3.3(a). In this case, the cell hit rate achieved was close to 62%, while it hardly reached the 43% when using a parametric-based model. As shown in Figure 3.3(b) successful cases rose to 70% and 54% using a histogram-based or parametric distributions respectively, when considering adjacent cells as good estimations.

According to these results, we decided to continue our analysis focusing on the histogram based approach. In that sense we notice that the shape of the histogram is particularly sensitive to the number of bins and therefore it influences the accuracy obtained. For that reason it required an additional analysis in order to find the right number of bins that should be used in order to get a better representation of the data modeled. There are several aspects that were taken into account before choosing the most appropriate value. If the bins are too wide, important information might get omitted and it would be difficult to find the underlying trend in the data. However, if the bins are too narrow, useful information cannot be extracted due to random variations that show up because of the small range at each bin.

Factually, there is no right or wrong answer about the most appropriate number of bins. Therefore, to determine whether the bin width was set at an appropriate size, different bin widths were empirically tested to analyze the sensitivity of the histogram shape. Figure 3.4 shows the results obtained using a histograms-based distribution model varying the number of bins. When a greater number of bins are used (e.g. 20 bins), accuracy shows a notable improvement reaching a 80% of cell hit (considering adjacent cells). We observe that a higher number of bins does not improve the system

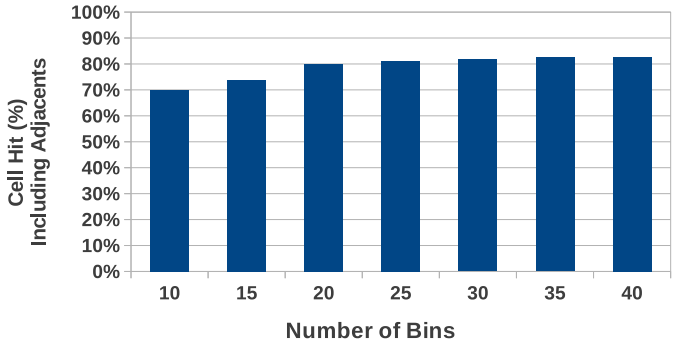


FIGURE 3.4: Accuracy obtained depending on the number bins used to build the histogram-based distribution. Percentage of successful cases includes adjacent cell hit.

accuracy notably. It means that it is not possible to extract additional information from these RSSIs to define their behavior more efficiently. Of course, the number of bins considered must be in accordance with the number of measurements available to build the histogram. It is worth mentioning that every bin has to contain at least one sample, in order to discard zero probability situations. Hereinafter, the main goal is to improve the results obtained using a histogram-based (20 bins) model.

3.5 Localization using spatio-temporal constraints

In the previous section we performed a set of experiments that allowed us to select the core technique which provides better accuracy results. However such evaluation was carried out without considering additional information in regards to the motion behavior patterns of the users at the scene. Each time we estimated the position of one device it could be localized at whatever position available in the scenario (among the different cells defined), since spatial restrictions were not imposed. For example, in two consecutive time instants separated by a few seconds, the mobile device could be localized at two different places spaced out by a distance larger than the physical space that could have been covered at a normal pace.

To solve this important issue, we decided to integrate a *hidden Markov model* (HMM), fully described in [105], since it was successfully incorporated in previous solutions [75, 115] where their authors demonstrated it usefulness as a good motion system model.

Given a user position, this method spreads the probability of moving over those points that are reachable during the next interval of time (considering a specific refresh rate). The performance that can be obtained will depend on the design of the Markov chain (or transition matrix), which encodes the assumptions about the transitions between each pair of cells. This is the critical point of using this HMM, the definition of this transition matrix, since it is necessary to take into account the normal mobility patterns of users around the scenario modeled.

Taking into account the information regarding the adjacency between the different cells defined, it is possible to build the transition matrix A where each element $a_{i,j}$ represents the probability of transition from the cell c_i to the cell c_j. This matrix can be automatically obtained by means of spreading the likelihood of moving from one cell to those which could be reached during the period between two consecutive estimations. Therefore, once we designed the matrix A considering the normal behavior of users around the scenario, as Equation 3.2 showed, being π the probability distribution vector over C, we defined $\pi' = A\pi$ as the probability distribution vector at the following consecutive instant time. To improve the system accuracy, we can define two different transition matrices, one of them representing the possibility of moving to a nearby cell when the device seems to be static, and another one more appropriate for motion situations in which there is a higher probability of transition between cells. In the first case, the possibility of remaining at the same cell is higher than the possibility of motion, in contrast, the second matrix has a higher likelihood of moving between cells.

Considering that, at this stage, we did not integrate inertial sensors into the system (like the accelerometer), the only option to estimate possible displacements was integrating WiFi measurements over time. In the literature we found different proposals in order to perform the motion estimation purely based on the analysis of WiFi signals. Krumm and Horvitz [113] measured the variance of the signal strength of the strongest AP to roughly infer whether the user was stationary or in motion. Another proposal was made by Muthukrishnan et al. [140], where authors presented the *spectrally spread motion detection* (SpecSMD) algorithm, an inference system based on euclidean distances between signals. In this case the authors observed that the signals' strength from APs appear to jump around more vigorously when the device is moving in contrast to when it is still, and the number of detectable access points vary considerably while the user

is in motion. We implemented this last technique, making use of a window time of the last n measurements to analyze the differences among the signals considered.

Figure 3.5 shows the accuracy obtained after repeating the previous tests but now integrating the histogram-based method with the HMM that has been defined. Using the motion inference technique we are able to roughly detect when the user is still or in motion, thus we can use the most appropriate version of the transition matrix A at each case. As we can observe, the integration of the HMM clearly improves the system accuracy since the cell hit rate achieved rises up to 90% in tested cases when the device was still.

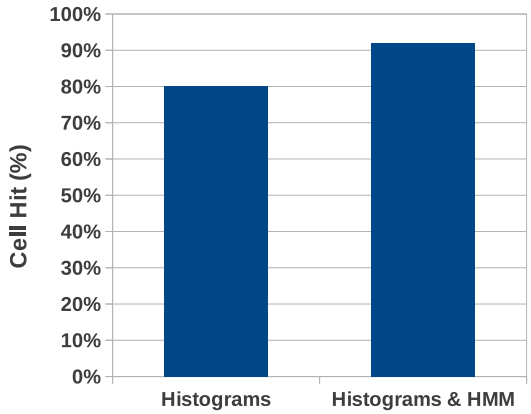


FIGURE 3.5: Accuracy (cell hit including adjacent) obtained using histogram-based and including the HMM algorithm.

We also tested the performance of this method while moving throughout the scenario. To carry out these tests we made use of a different validation dataset of measurements. It was obtained using the training device but with the exception that in this case we covered the path indicated in Figure 3.6. Due to the delay introduced by the passive scan, the motion pattern was set to allow the collection of one measurement at each cell. This test was performed several times using each one of the different system models. As we can observe in Figure 3.7, in comparison with previous results, accuracy is notably decreased when user is in motion, falling down to 73%.

These results may be negatively influenced by using a single measurement to estimate the position. In this case, it is possible to lose the signal reception from any AP but also a bad RSSI obtained from a reflected signal may be used to calculate the position. In these situations, the method is adversely affected and bad estimations may be obtained. We can minimize these negative effects by using a window of time to consider all

measurements inside when estimating the current position. The signals collected from the same AP can be averaged, and more reliable measurements will be provided to the localization engine. In scenarios like ours, where a high-density layout was defined, the maximum time available to estimate user position while moving from one cell to another is reduced because of the proximity between cells, just 1 second. Additionally, using devices not able to perform active scans, and considering the high delay introduced by the passive scans, discards this possibility. However, there are other situations in which this approach could be introduced, as for example, when WiFi is used to perform coarse-grained estimations in more spaced out layouts, where the time needed to move from one cell to another is greater than the scan delay of the devices.

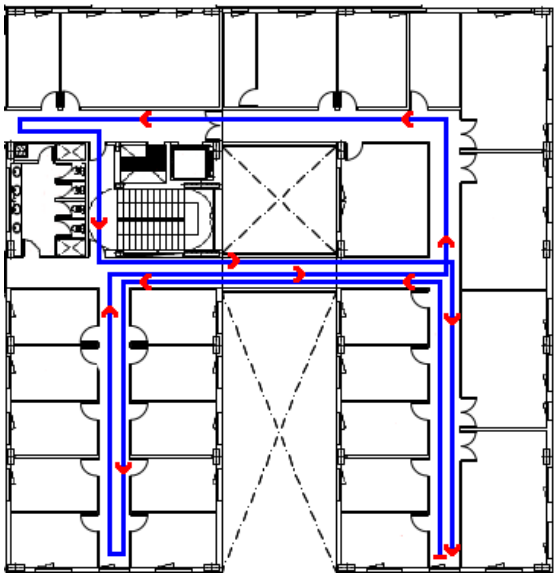


FIGURE 3.6: Path followed during motion test.

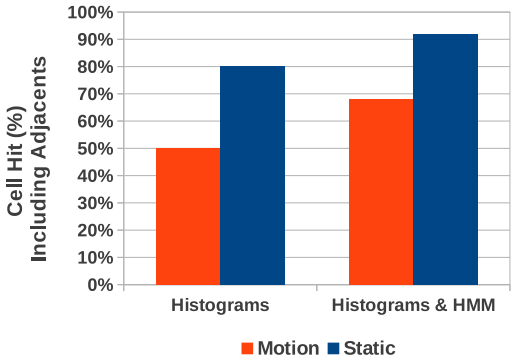


FIGURE 3.7: Accuracy (cell hit including adjacent) obtained using histogram-based and including the HMM algorithm in motion tests. Static results included for comparison.

3.6 Localization supported by contextual information

In some situations we can take advantage of contextual information to improve the system accuracy, for example, using users profiles to establish certain restrictions to get access to some dependencies, offices or labs. These restrictions are related to each device's owner according to their specific role within the scenario in question. For example, in a hospital environment, patients and visitors have restricted access to some dependencies like laboratories or surgery rooms. Taking advantage of this a priori knowledge, and considering that each user has a specific role (patient, visitor, staff), it is possible to take into account these restrictions when estimating its position. In consequence, we could discard those areas (set of cells) where some devices cannot be located, or at least, where they have a lower probability to be located. Apart from obtaining better accuracy, in some cases it implies a major reduction of the search space.

Following this idea, our proposal requires the classification of each room according to its access level, as well as the classification of the device's owner in one of the possible profiles. This task can automatically be done by considering the identifier in the hypothetical case in which users use an application that requires to be logged-in to access to additional services. In the case of the university environment, email address can be used to identify the user, allowing to differentiate between students, professors and staff. Additionally, the classification of users can also be done by analyzing the user behavior during a period of time. In [152] we proposed a novel technique able to classify people moving around a hospital by analyzing their behavior. We considered the time that users stay inside the hospital or the places they used to visit, among other information, to classify the users. Then, after a period of time, once users are automatically labeled, restrictions can be applied during the localization step.

In order to assess our proposal, we conducted some tests within our testbed scenario assuming two different user roles: student and professor. Students mainly move along the corridors and laboratories, so the cells belonging to these dependencies were labeled as *public* cells. On the contrary, the rest of dependencies (those defined mainly inside professors' offices or meeting rooms) were labeled as *private*, and only university staff and professors can gain access to them. Obviously, students could enter a professor's office for tutorials, for that reason zero-probabilities are avoided, considering a lower probability of localizing a student at those dependencies than in the case of a professor.

Consequently, here we propose the so called optimization *Path-Restricted Location (PRL)* with the aim of minimizing the number of cells where users can be located and reducing the probability of accessing some of them according to their profiles. The mentioned restrictions were encoded in the HMM transition matrix. We tested this proposal applying the same dataset as in the motion experiments presented in previous section. We assumed the device was carried by a student and therefore he had limited access to private rooms. The same path indicated in Figure 3.6 was covered during these experiments. Since it took place mainly at corridors, there were a minimum possibility of obtaining wrong estimations inside the rooms. As it can be seen in Figure 3.8, PRL improves the accuracy achieving a cell hit rate near 73% in comparison with the 68% previously obtained.

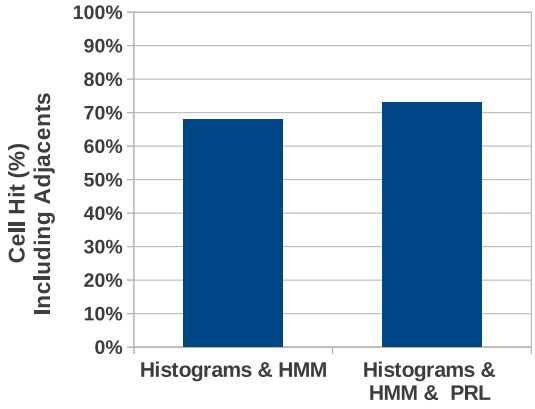


FIGURE 3.8: Accuracy obtained (cell hit including adjacent) integrating the PRL method within the definition of the HMM transition matrix.

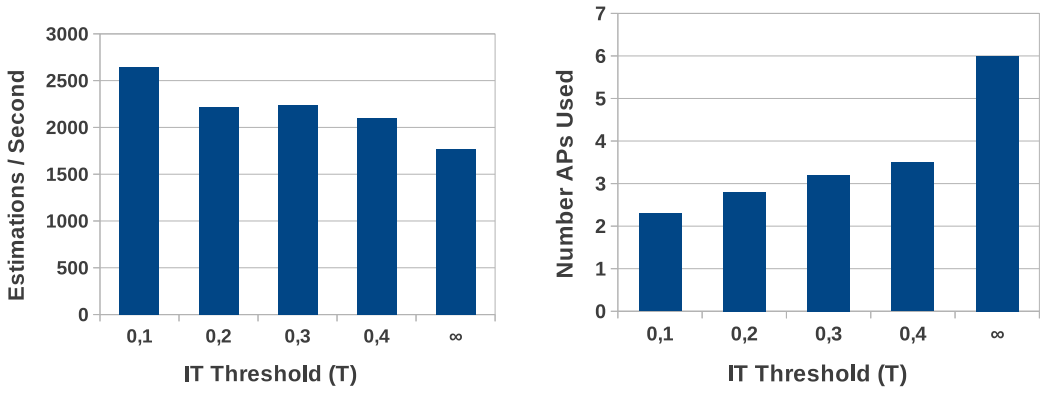
3.7 Improvement of the system scalability

In this section we summarize different proposals to address the important issue of the scalability. On the one hand, some of them were focused on obtaining the best possible accuracy but at the same time trying to reduce the number of operations to perform each estimation. On the other hand, we found other solutions based on the analysis of signals' behavior to define coarse-grained layouts that boost the probability of success, though at the expense of losing accuracy.

3.7.1 Optimizing the estimations without affecting the accuracy

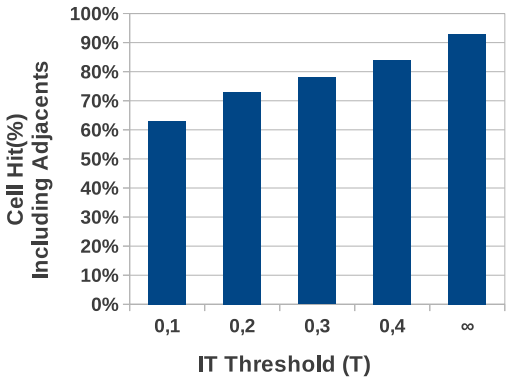
Focusing on the first kind of solutions, the scalability issue can be alleviated by reducing the computational cost of performing the location estimations minimizing the number of operations to be done. It is possible to estimate a greater number of locations per second while supporting a higher number of users. Previous contributions like the one proposed by Youssef et al. in [193] discussed several interesting techniques to be integrated within our current solution to address this issue. Among the different proposals that authors made, we want to focus on the *Incremental Triangulation* (IT) clustering technique. It is based on the simple idea that the strongest signals come from the nearest APs, and therefore we can assume that those signals are more stable and more reliable. Thus, when estimating the location of a device using the received signals ordered by their intensity, these signals are evaluated according to their reliability. During the location estimation process the APs are used iteratively, starting the location process using the strongest received one and repeating the process by adding more APs until a good estimation is obtained or there are no more APs that can be used. If the probability assigned to the estimated location is significantly higher than the probability of the second most probable location (up to a specific threshold value T), the most probable location will be considered as the final location estimation, and the rest of the available APs are not taken into consideration. T is considered as the difference ratio between the most probable and the second most probable position.

As a consequence, it is possible to reduce the time required to perform each individual estimation, and to process more operations per second (see Figure 3.9(a)), obtaining up to 18% of additional estimations when $T = 0.4$ in comparison with the case described in Section 3.5, in which we did not use this optimization (right column in Figure 3.9(a)). This improvement comes from the reduction of the space search, which is now restricted to those cells covered by the used APs. Figure 3.9(b) shows a comparison of the averaged number of APs used during the location estimation process, considering different values of T. Finally, as we can observe in Figure 3.9(c), using $T = 0.4$ we are still able to obtain a 83% of successful cases in our tests, which can be an acceptable solution for those cases in which we are interested in getting a coarse-grained location estimation. We consider that the effects of this optimization could be observed better in larger environments with



(a)

(b)



(c)

FIGURE 3.9: Performance analysis varying the difference ratio T between the most probable two positions. a) Number of estimations performed per second; b) Number of APs considered; c) Percentage of cell hit achieved (including adjacent) using Histograms+HMM+IT.

a huge number of APs, like an airport or a hospital, where it can mean an even greater time reduction.

3.7.2 Improving the scalability by reducing the granularity

On the other hand, Lemelson et al. proposed in [119] the *fingerprint clustering* algorithm to estimate the position error that is inherent in 802.11-based positioning systems. It makes use of the training RSSIs to find clusters based on the idea that the signals collected in nearby cells tend to cover only a limited range of the possible values. In such a case all the fingerprints collected in one office room are very similar, and a positioning algorithm will hardly be able to make an exact position estimation. Instead,

it will probably select one of the other fingerprints collected somewhere else in the room. Following this approach it is possible to define clusters made up by several adjacent cells in which signals have similar properties, using these clusters to build a coarse-grained fingerprinting map. This new map can be used to build the system model, which gives more reliable estimations at the expense of reducing the accuracy.

The technique used to find the set of clusters is the following. Initially each cell represents a single cluster containing all the training samples collected inside the physical area of the cell. For each AP observed in two adjacent clusters, it is calculated the intersection area of the two normal distributions obtained from the mean and the standard deviation of the measured signal strengths obtained at each one of them. Then, the similarity measure for a pair of clusters, which indicates whether these clusters must be joined, is estimated as the average size of all intersection areas computed from common APs. If the obtained value is up to a specific threshold, both clusters are considered similar and they are joined into a new one. The process is repeated until no more clusters are joined. The evaluation of the similarity of two clusters is subject to the adjacency condition between them. That is, two non-adjacent clusters will not be considered as candidates to be joined. Unlike other classical clustering techniques, like k-means, in this case we do not have the possibility to indicate the number of clusters to generate. On the contrary, the final clusters' distribution will depend on the signals' behavior and the similarity threshold value, which should be evaluated empirically at each different scenario.

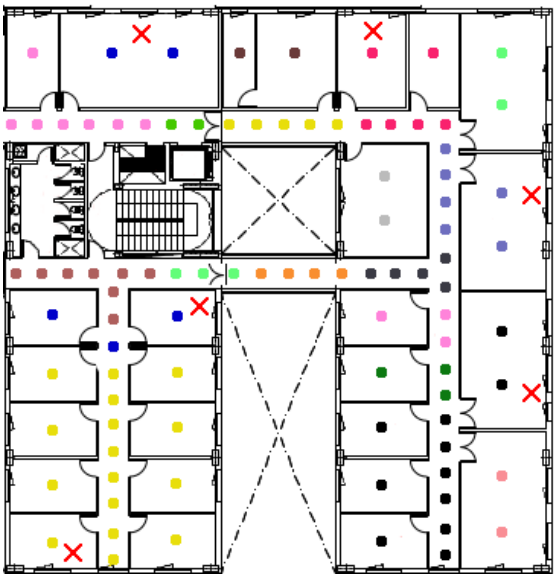


FIGURE 3.10: Clusters obtained from RSSIs analysis.

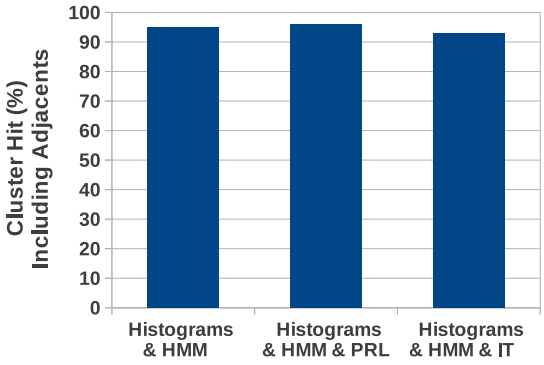


FIGURE 3.11: Cluster hit probability using the different techniques described.

In order to show how this proposal can be applied, Figure 3.10 depicts the different clusters that were obtained after analyzing the training RSSIs. Then, using this clustering-based spatial distribution, Figure 3.11 shows the cluster hit obtained using the different techniques that have been mentioned previously. Most of the already analyzed techniques obtain a high cluster hit percentage, up to 93%. Therefore, we can conclude that the proposed clustering technique provides a high reliability to perform coarse-grained estimations. As it will be described in Chapter 4, we can manually join the obtained clusters in order to define medium-sized zones, according to the physical distribution of the environment. Then, using these new defined zones, it is feasible to estimate a coarser estimation of the device location within any of them with a higher probability of success, where WiFi-based location might be further refined using other sensors. This will be a good starting point for the multisensor integration, since this WiFi-based estimation can be used to reduce the space search when analyzing additional sensor data, like images, thus improving the system performance and scalability.

At this point, we want to mention that there are other approaches [118, 180, 189] in order to model larger buildings and define larger zones considering the logical distribution of the environment. These solutions are less intensive in terms of training effort, that is, it is not necessary to carry out such intensive training processes all around the scenario. However, with the technique we integrated, we want to emphasize the importance of performing a clustering guided by the efficiency of the obtained signals with the aim of achieving the desired accuracy.

3.8 Support for heterogeneous devices

Besides accuracy, scalability and performance, one of the imposed requirements of every location-based system is the support of heterogeneous hardware clients. Due to the wide range of devices on the market nowadays, the functioning of these systems cannot be restricted to specific hardware. However, different devices provide different intensity readings, depending on their antennas, the transmission power and other factors, having negative implications over the final system accuracy. For these reasons, we focus in this section on the importance of carrying out a calibration process in order to ensure an appropriate functioning of the system independently of the device characteristics. We initially focused on the integration of heterogeneous devices based on the same radio technology, in this case WiFi, and after that we will define how we can make our system compatible with other radio technologies, like Zigbee.

3.8.1 Integrating heterogeneous 802.11 devices

One of the existing alternatives is the one proposed by Gwon and Jain in [74]. They proposed a calibration-free location algorithm that eliminates off-line RSSI measurements and makes use of a *Triangular Interpolation and eXtrapolation* (*TIX*) algorithm to estimate the device location. One of the main disadvantages of this proposal is that it requires to know the position of the APs used to build the model. This requirement is not feasible in large scenarios where we cannot control the wireless infrastructure. In [189] authors proposed a calibration-free algorithm, FreeLoc, which addresses the heterogeneity of devices estimating their positions based on the relative order of the received signal strength. We had the opportunity of testing this technique, using as an experimental environment the entire Faculty of Computer Science, obtaining good results in terms of feasibility but using a coarse-grained grid layout. This proposal seems to be appropriate for those cases in which obtaining a high accuracy is not an important requirement and it is not possible to carry out an exhaustive training.

Another solution is the one made by Kjærgaard and Munk [102]. They proposed a *Hyperbolic Location Fingerprinting* (*HLF*) which addresses the heterogeneity of devices by considering signal-strength ratios between pairs of base stations instead of absolute

signal strengths. Then comparing these computed signal-strength ratios with the fingerprinted ones it is possible to estimate the device position. Furthermore, Kjærgaard [101] also proposed another technique which tries to automate the calibration process based on movement detection, to group the same-place measurement into calibration fingerprints.

Haeberlen et al. in [75] proposed a calibration function based on the linear relationship indicated in Equation 3.3, where i represents the signal intensity value observed by the device and c(i) the value that would have been observed by the training device.

$$c(i) = c_1 \cdot i + c_2 \tag{3.3}$$

Computing the least-squares fit between the observations obtained by the new device at some specific cells and the corresponding values from the sensor map, it is possible to obtain the parameters c_1 and c_2. This method requires the definition of at least two different calibration points (cells in our map), where the devices must collect geo-tagged RSSIs to run the process. During our test we made use of three calibration points to over-determine the equation system that estimates the parameters c_1 and c_2.

In order to analyze the implications of the calibration over the final accuracy, we carried out several experiments using the Haeberlen[75] manual approach to calibrate different devices and evaluate whether this adjustment influences the final accuracy obtained. The devices used during the experiments were hand held and inside the pocket, with no meaningful differences in the final results. Figure 3.13(a) shows the RSSIs received during one minute from one of the APs observed. Unadjusted RSSIs clearly do not fit to those observed by the training laptop, which was used to build the fingerprinting map. Nevertheless, once they were calibrated, see Figure 3.13(b), signals present a similar behavior, having a positive impact over the final accuracy.

In previous sections we presented empiric results that were estimated by using as input the observations collected during the training phase and applying a 4-fold cross validation method. Nevertheless, we also performed several real time tracking tests using different devices and different motion patterns. We performed different tests, both remaining static at several points of our scenario (including corridors, offices and labs) but also

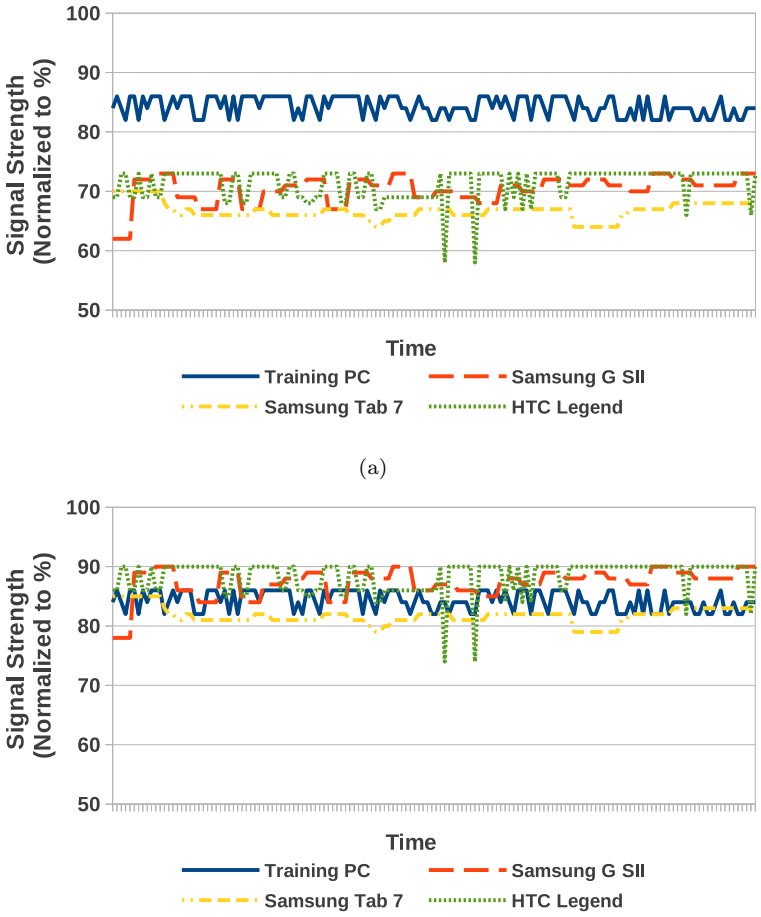
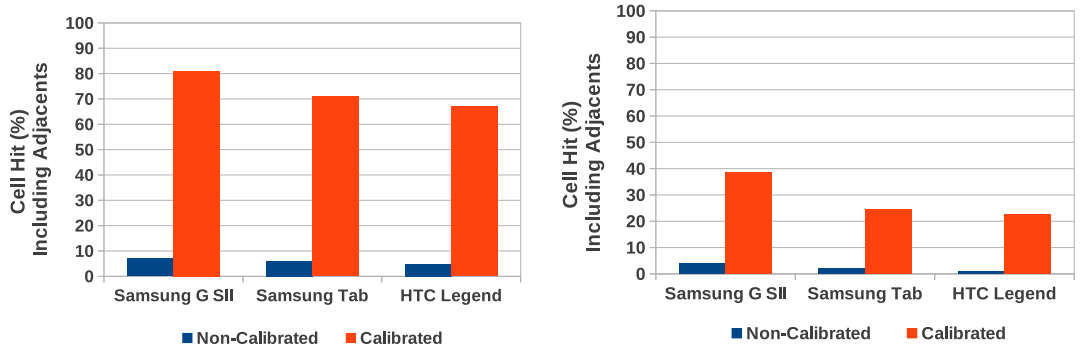


FIGURE 3.12: (a) RSSI observed by different WiFi devices before their calibration. (b) RSSI observed by different WiFi devices after their calibration. Values normalized to % considering the maximum and minimum reception power (dBm) of each device.

using a motion pattern while moving mainly along the corridors, where several rounds were performed.

During the motion tests we ensured one observation, at least, at each cell along the covered path. Figure 3.13 presents the averaged results obtained during both static and motion test, considering the case in which devices were not previously calibrated and also the results after their calibration. From the results we can conclude that we are able to get a relatively good accuracy when locating static devices, reaching a 72% of cell hit. Nevertheless, this accuracy decreases when trying to estimate the position of a device in motion, obtaining only a 30% of correct estimations. From these results we can derive that there are important difficulties when trying to use a WiFi-based localization system to obtain the position of a device with high accuracy, that is, obtaining a maximum estimation error down to 3 meters.



(a) (b)

FIGURE 3.13: (a) Cell hit achieved (including adjacent) using heterogeneous devices during static experiments. (b) Cell hit achieved (including adjacent) using heterogeneous devices during motion experiments.

3.8.2 Integration with other radio technology (Zigbee)

As we mentioned at the beginning of this section, we introduced the possibility of integrating the use of heterogeneous devices based on different radio technologies. In that sense, we focused on the use of 802.14.5 Zigbee technology to perform our experiments, since we found it very useful from the point of view of localizing some devices which do not count with WiFi connectivity, as for example, some of the equipment present in the hospital environment. Due to the reduced size and the long battery life of a Zigbee device (or mote), we can estimate the position of other elements just by attaching them a Zigbee device.

The main goal was to test whether we are able to estimate the position of Zigbee devices avoiding new training processes to build Zigbee specific fingerprinting maps, that is, using the already obtained WiFi-based system models. To carry out this integration we based on the possibility of adapting the reception pattern of the signal strengths of Zigbee devices to the one observed by the WiFi ones within the same environment. The idea behind this proposal is to calibrate the Zigbee device applying the same method that was used to calibrate the WiFi devices, but using a specific wireless network based on this new technology.

We placed different Zigbee motes (*emitters*) at the same positions as the WiFi APs were placed (see Figure 3.15). Using this similar distribution, we wanted to simulate the same scenario deployed for the WiFi case, with the exception of the differences in the power of transmission between the emitters and the APs. The distribution of these emitters

allowed the reception of at least three of them in almost 90% of the defined cells. As we can observe in Figure 3.15, due to lower transmission power of these emitters, there were some cells in which a fewer number of emitters were detected.

In consequence, the existing WiFi-based fingerprinting map can be used during the localization stage to estimate the position of the Zigbee device. Figure 3.14 shows the averaged RSSI values received by a WiFi device or a Zigbee mote at each cell located at corridors. We considered the averaged value of 20 different measurements normalized as percentage, that is, taking into account the maximum and minimum reception capability of the mote. The green-square line represents the RSSIs observed by a WiFi device from one AP. The blue-diamond line shows the values received by a TelosB Zigbee mote from one emitter (the one placed at the same position as the previous AP) when no calibration method is applied. As it was expected, the received intensity values clearly differ from those obtained using the WiFi device. Nevertheless, after calibrating the mote (red-triangle line), both patterns are similar. Within our environment, the calibration parameters were established to $c1 = 0.6$ and $c2 = 49.5$, see Equation 3.3. Similar results were obtained independently of the AP and its correspondent Zigbee emitter considered.

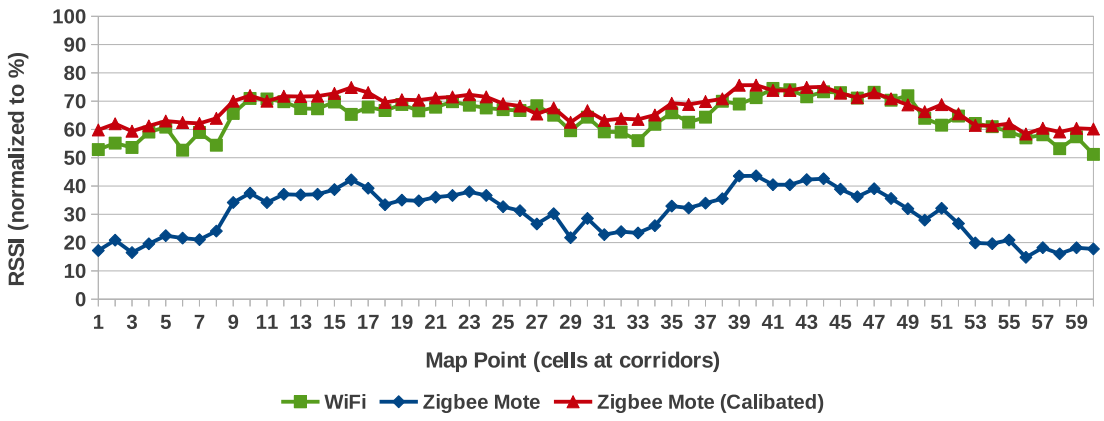


FIGURE 3.14: Calibration results using different technologies for one AP in our scenario. Similar results were obtained for the rest of APs.

In order to complete the entire evaluation of this approach, we built a fingerprinting map of Zigbee signals. Once the new system model was obtained, we performed different localization tests using both the WiFi (calibrating the RSSIs obtained by the mote) and the Zigbee-based (without modifying the RSSIs) systems models. During our tests we made use of a Zigbee mote while moving through the scenario, following the path indicated in Figure 3.6. The mote was calibrated as it was indicated previously, using

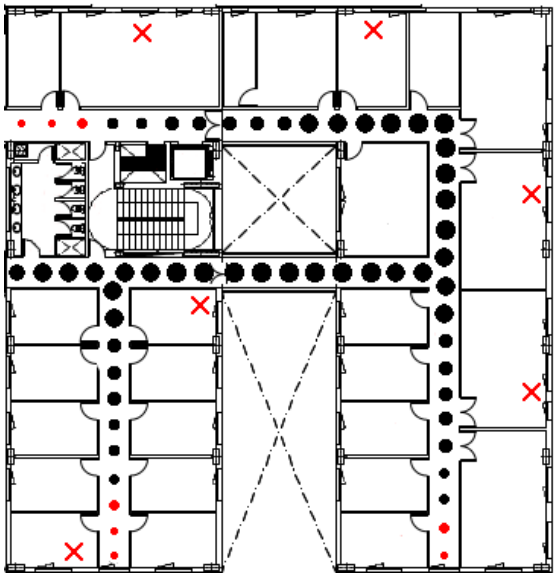


FIGURE 3.15: Map of coverage of Zigbee emitters. Red crosses indicate the position where emitters where placed. Black dots represents the observation of at least 3 different emitters. Red dots represents those cells where less than 3 emitters are observed.

the wireless Zigbee infrastructure. As we can see in Figure 3.16 there are no important differences on the accuracy obtained independently of the system model used. In our test we obtained around a 33% of successful cases making use of the WiFi-based system model, whilst using the Zigbee-based system model the cell hit was around 37%. The results clearly fit with those presented in previous section 3.8.1, and despite the low accuracy obtained they confirm the compatibility between radio technologies. We consider that this opens new opportunities when deploying other proposals based on a mixture of radio technologies.

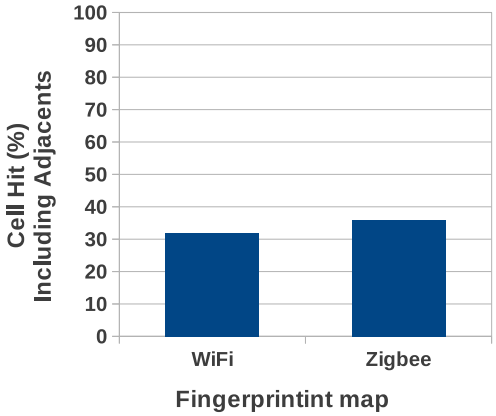


FIGURE 3.16: Location estimation results (in terms of cell hit including adjacent) of a TelosB Zigbee-based mote using a Wifi-based or Zigbee-based fingerprinting map.

3.9 Summary

Throughout this chapter we have carried out an in-depth review of the different localization techniques already present in the literature, accomplishing different experiments suitable to assess those considered more useful for our interests. This analysis was made taking into account that our goal was to study the convenience of using the WiFi technology for localization, with the purpose of the integration of additional sensors that will provide an added value to our estimations.

We have focused on those techniques based on the fingerprinting methodology because of their suitability, both in terms of the accuracy and the performance provided in previous works. Integrating several of the described techniques, we developed a localization system providing a good performance according to the different results obtained. Moreover, we introduced the possibility of improving the system performance making use of additional contextual information. We are aware that this technique must be refined in order to fully take advantage of it, for example, carrying out an automatic categorization of users and performing a better analysis of motion patterns, amongst others.

We analyzed different techniques that allowed us to improve the system scalability, either with a view to refine the system performance providing a certain level of accuracy or by means of integrating clustering techniques which provide high reliability at the expense of obtaining less accuracy. Consequently coarse-grained localization estimations can be obtained, and as we will describe in the next chapter, the use of this algorithm will be very beneficial for the multisensor integration.

The use of calibration techniques supports the compatibility of heterogeneous devices within the system. We demonstrated that it is possible to make use of WiFi devices with different characteristics, but also the utilization of devices based on other radio technologies. The possibility of making use of different radio technologies, like Zigbee, within the same scenario avoids the necessity of creating specific fingerprinting maps for each one of them. This is an advantage with regards to the possibilities of tracking those devices which are not equipped with WiFi connectivity, for example attaching to them a Zigbee mote, which according to its characteristics (size and battery life) may support their localization.

To summarize, the main remarks extracted from this study are:

- We evaluated different techniques within a real scenario, analyzing the usefulness of the WiFi technology to deploy location systems for indoor environments.
- We realized the importance of the calibration process, since the accuracy obtained after the calibration of the device clearly increases the percentage of successful cases.
- It is important to mention the notable differences that exist between the accuracy obtained when the devices are mainly stationary in comparison to the one obtained when localizing devices in movement, what enforces the necessity of using other sensors to improve the system accuracy and reliability.
- The results demonstrate that in spite of being a lightweight technology (at least in computation time) it lacks efficiency when it is required to provide accurate localization solutions. However it is appropriate to provide coarse-grained estimations making use of the alternatives based on clustering.

To conclude, in the global context of this thesis, the work described in this chapter supposed an initial step that allowed us to value the different alternatives that WiFi provides. We analyzed a wide variety of proposals providing fine-grained (around 3 meters) and coarse-grained (more than 3 meters) accuracy relaxing the training conditions and providing a higher reliability. Because of the intrinsic behavior of the WiFi signals, we concluded that there was no other alternative to provide a better localization accuracy. In consequence, in the following chapters we will evolve our system by integrating additional sensors, using the WiFi as the keystone which facilitates their integration.

Chapter 4

Positioning based on computer vision techniques

4.1 Introduction

In the previous chapter we carried out an intensive analysis demonstrating the usefulness of fingerprinting techniques using 802.11 signals to deploy localization services. Despite the good results in terms of accuracy (around 2.5 meters of average error), which could be precise enough for some services, there are other situations where a better accuracy is necessary to support those applications with higher requirements.

Better results can be obtained by integrating the information captured by multiple sensors. Specifically, localization systems might benefit from the images that users obtain from the camera, as for example in order to display augmented reality in certain applications. Those images can be used to provide a better estimation of the users' position, offering a seamless integration of the sensor and the display.

In this chapter we present two different proposals regarding the integration of computer vision techniques for image analysis. One of the main characteristics of these proposals is their non-intrusive nature, that is, no additional elements (landmarks or objects) are required to be installed in the environment, what makes them even more interesting compared to other existing approaches. In both cases, we made use of the *Scale Invariant Feature Transform* (SIFT) [127], an image processing technique suitable for image

matching and object recognition, which allows the extraction of distinctive features from the images captured by the device's camera.

Our initial approach can be categorized as a *place recognition based* technique. It integrates the image analysis within the probabilistic model based on RSSIs that was defined in the previous chapter. The current method is based on the creation of a fingerprinting map of WiFi RSSIs and feature descriptors extracted from representative images of the application environment. An initial coarse estimation, based on IEEE 802.11, is accomplished to determine a cluster of physical cells, or zone, where the device seems to be according to the received signal strength. Next we carry out an image analysis to look for coincidences between the current image and those stored in the database. Taking advantage of the initial coarse estimation made, we can constrain the space search to those images acquired at cells belonging to the tentative zone. Nevertheless, one of the main drawbacks of this solution is that it is limited by the granularity of the sampling procedure, that is, the estimation result corresponds to the location of the cell which provides more matchings with the current image. As we will show, the power of this multisensor approach is demonstrated, since 95% of the estimations are exact or directly adjacent to the right cell.

However, there are several applications which require a higher accuracy. For that reason, we extended this solution and, in the second proposal, we made use of *Structure from Motion* (SfM) techniques to build off-line 3D reconstructions of the scenario from the correspondences among SIFT descriptors of training images. Despite the added complexity of the off-line phase, this approach implies a notable evolution of our system from the point of view of the accuracy provided. In this case, we assumed that the users are capturing images, typically with a tablet or a smartphone, for which they want to obtain the precise estimation of the full 6 degrees of freedom (dof) – 3 for (X, Y, Z) position and 3 additional for the (α , β , γ) Euler angles for the rotation matrix – of the device with respect to some conveniently chosen world reference coordinate system. Several resection techniques were tested to estimate the full 6 dof, depending on whether we know some intrinsic camera parameters, like the focal length. As in the previous case, despite of the fact of using the images obtained from the camera as the main piece of data to infer the position of a particular device, we can benefit from other sensors, like WiFi, accelerometer or compass, in order to constrain the space search. The multisensor proposal described is a valuable contribution for this kind of location-based services,

since we have obtained a good trade-off between a fine-grained accuracy (around 12 cm of average error) and an acceptable response time (around 250 ms) integrating several sensors. This solution is able to support a wide variety of augmented-reality applications which do not require a real time response, such as those designed to display location-aware reminders or notes, or to obtain information about who is behind the door of a particular laboratory.

The rest of this chapter is structured as follows. Section 4.2 describes our initial proposal, providing additional information about the different experiments carried out and the obtained results. Afterwards, in Section 4.3 we describe the proposal regarding the use of 3D models and camera resection techniques to estimate the current 3D position of the device. A detailed analysis of the accuracy and performance of the solution is included. Finally, Section 4.4 presents our main remarks.

4.2 Positioning based on image recognition

In this section we introduce our initial solution derived from a classical fingerprinting map based on RSSIs which also uses SIFT descriptors from images obtained at different training points. Using these feature descriptors extracted from the images captured using the smartphone camera, we perform a matching process against thousands of features previously extracted from geo-tagged images to determine the current location of the device. Within computer vision research field, this type of approaches are well-known as *place recognition based* solutions [173].

Figure 4.1 summarizes the overall functioning of this proposal. As it is required for fingerprinting based solutions, an initial training phase is carried out. During this stage we analyze the RSSIs (Step 1 in Figure 4.1) to build fingerprinting maps and to estimate the clusters where signals have a similar behavior. The images are also processed to obtain their features (Step 2). Using clustering information of RSSIs, we build the image search structures (Step 3) that will be used during the on-line phase.

Images are the main piece of data to infer the position of a particular device, but we can benefit from other sensors in order to limit the amount of information to be examined. During the on-line phase, using fingerprinting methods based on the RSSI of transmitted 802.11 packets, we can obtain an initial estimation indicating the cluster of physical

points where there is a high probability of locating the device (Step 4). This process may make use of information from the accelerometer when it is available. Images are filtered according to their quality (Step 5). Then features obtained from current image (Step 6) are used as input for the image matching process (Step 7). Using the initial estimation we can reduce the amount of images in the database to check against, since only those images contained in the selected cluster are analyzed.

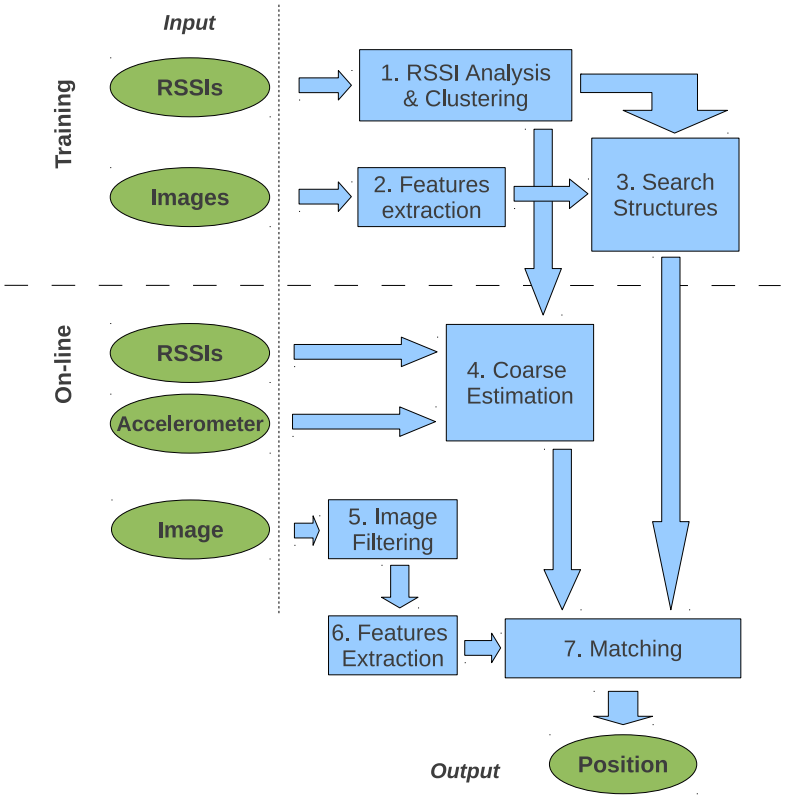
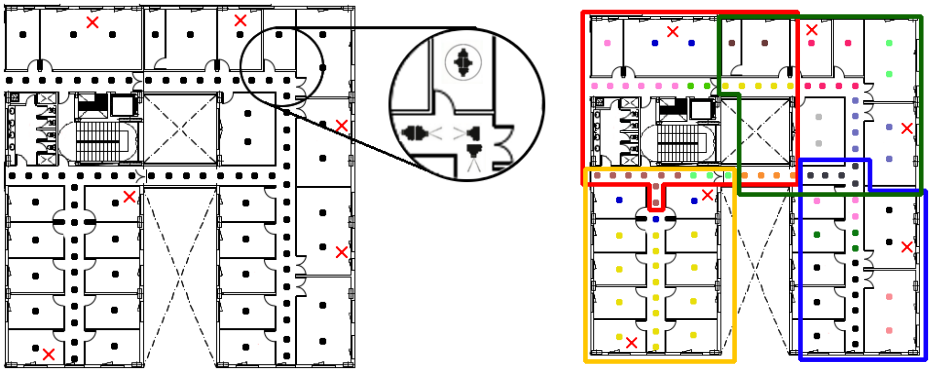


FIGURE 4.1: Overview of the place recognition proposal. It includes all the steps to be taken during the training and on-line phases.

4.2.1 Experimental environment

For a better comprehension of the rest of the chapter, here we describe the scenario where the experiments mentioned throughout this section were performed. The testbed corresponds with the scenario described in Chapter 3. The novelty now is that we obtained a set of seed images captured in a different way at each cell depending on the cell type (zoom on Figure 4.2(a)). Thus, a cell in a corridor is associated to as many images as possible directions of movement, whereas inside the rooms we built panoramic images covering the whole dependency. Not all rooms were photographed because of privacy concerns.



(a) (b)

FIGURE 4.2: Experimental environment: a) Representation of image collection process. At each cell, we collected as much pictures as possible movement directions. Within labs, panoramic pictures were obtained; b) Clusters definition. Equally colored dots represent cells pertaining to the same cluster. The four colored "rectangles" represent the different zones which have been defined for image analysis.

Making use of the RSSI clustering techniques we calculated the clusters shown in Figure 4.2(b) (colored dots represent each cell of the discrete grid). Cells pertaining to the same cluster are displayed using the same color. Then, we defined four overlapping zones joining adjacent clusters. The definition of the number of zones and their sizes could vary according to design preferences, but always considering that cells belonging to the same cluster must be kept in the same zone. These zones were used in order to reduce the search space when images are analyzed.

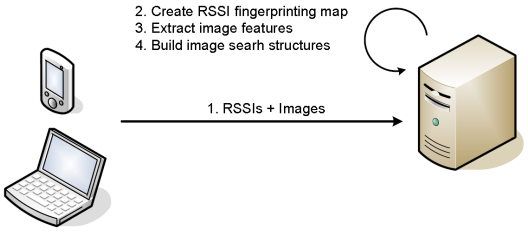
We defined a client-assisted architecture that distributes the processing responsibilities between the mobile devices (laptops, smartphones and tablets) that were used to carry out training task (but also will be the object of localization), and a core server in charge of processing the training data and carrying out the position estimations.

Our server was an Intel® Pentium® Dual-Core CPU E2160 server. For image processing, a nVidia GeForce 9800GT was installed, supporting both *OpenGL Shading Language* (GLSL) and *Compute Unified Device Architecture* (CUDA). We made use of the SIFTGPU library, a GPU Implementation of SIFT by Changchang Wu [9] for image features extraction.

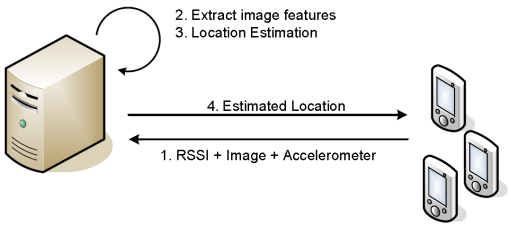
During the training phase (see Figure 4.3(a)), RSSI observations from the six APs deployed were collected with an Asus Eee 1201 laptop with a Realtek TRL8191SE WLAN 802.11n card. In this case, in order to save deployment time, we reused the RSSIs previously collected for experiments in Chapter 3. Training images were captured using

an HTC Desire smartphone. Image resolution was 640×480 pixels for corridors and 2900x360 pixels for panoramic images of offices and laboratories. This information was processed by the server in order to build the WiFi fingerprinting map, to estimate clusters based on RSSIs analysis and to perform the features extraction from training images. These features were used to build the search structures that were used to perform the posterior matching with the features extracted from images collected during the on-line phase.

Through the online phase (see Figure 4.3(b)) the system was tested using different smartphones and tablets based on Android OS: Samsung Galaxy SCL and Samsung Galaxy SII smartphones with Android 2.3.5, Samsung Galaxy Tab Plus 7.0 with Android 3.2 and ASUS EEEPad Transformer TF-101 tablet with Android 4.03. We developed an application that was used to collect data from different sensors (WiFi, accelerometer and camera) and to send the measurements to the server. This application ran in background, populating the server repository with this information, which was used when other services required the position of the specific device. A simple augmented-reality application was also implemented in order to test the system accuracy. This application constantly performed localization queries to the server in order to obtain the current position of the device. For its use, operators hold the phone out in front of them, facing ahead in order to obtain location-aware information related to their current position.



(a) Training phase



(b) On-Line phase

FIGURE 4.3: a) Training phase; b) On-line phase.

We are aware that the use of these augmented-reality applications based on images would be sporadic, since we do not envision realistic scenarios where users are comfortable holding their phones facing ahead all the time while moving. For that reason, when the smartphone was inside the pocket, facing the floor, or obtaining useless images (out-of-focus or uniform), the location estimation was based mainly on the RSSI measurements. In section 4.2.6 we will show different experiments using several sensor sets.

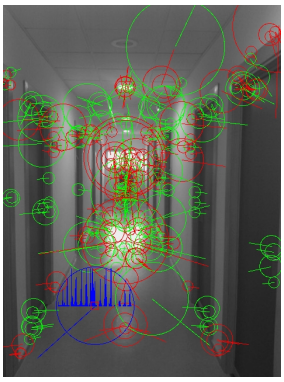
4.2.2 Scale invariant feature transform (SIFT)

Before getting started with our proposal, we will make a brief introduction describing the SIFT technique [126, 127], since it will be the keystone in the remaining work. SIFT is a widely adopted technique in computer vision research. It provides a method for extracting a collection of *visual features* from images, which are invariant to image translation, scaling and rotation, and partially invariant to illumination changes and affine distortion or change in 3D camera viewpoint. Each of these features are well localized in both the spatial and scale domains of the input image: they are characterized by a 4-D vector (x, y, s, θ), where (x, y) is the position in the image, s is the feature scale (i.e. size), and θ is a dominant orientation.

One of the most important advantages of SIFT is that once each feature vector has been correctly localized, it is subsequently characterized by an additional highly distinctive 128 dimensional description vector, which depends on the overall photometric structure of the local image environment of the feature. This descriptor allows a single feature to be correctly matched with a high probability against a large database of features, which provides a powerful basis for visual recognition. This fact has been exploited both in topological place recognition systems as well as in the context of simultaneous localization and three-dimensional scene reconstruction [63, 156]. The strictly local nature of the computation for each feature, together with the possibility of globally computing consistencies among the whole list of the obtained features, makes this technique rather robust to problems like occlusion, clutter, or moderate image noise.

We want to emphasize the power of SIFT features to provide a good distinctiveness even in environments with many similarities between different physical emplacements. In order to illustrate this characteristic, Figure 4.4 shows different images representing

the features extracted from one image captured at our testbed scenario, and the corresponding matchings found when comparing similar images. In Figure 4.4(a) each feature is represented by a circle indicating the corresponding scale and a vector showing the orientation (red circles are light blobs on dark background, green ones are dark blobs on light background, and a specific 128-dimensional feature is drawn as a blue histogram, for illustration purposes). As shown, though corridors A and B are clearly very similar, there are scene details that still make a significant difference regarding the number of features matching with the captured image when the image corresponds to the same scene 4.4(b) or not 4.4(c).



(a)

(b)

(c)

FIGURE 4.4: a) SIFT features detected at one image; b) Matchings found when comparing two different images captured at corridor A; c) Matchings found when comparing two different images captured at corridors A and B, respectively. Despite they are very similar, fewer correspondences are found.

Figure 4.5 summarizes the main computation stages that take place during the process of obtaining the collection of features from an image. This process is divided in two main phases. The first phase consists of an extrema detection in the laplacian of gaussian scale-space of the input image, which has to carry out a computationally intensive iterative filtering, scaling and differentiation of the input image (Steps 1 and 2 in Figure

4.5). The laplacian of gaussian is in fact approximated by a difference of gaussian procedure, which together with the downscaling of the input image for larger scales allows to partially alleviate the computational burden of the involved image processing needed. The potential points of interest are then obtained by performing a global search over all scales and image locations (Step 3). The list of obtained features is finally polished removing unstable and poorly localized points, as well as performing a final interpolation procedure to get subpixel accuracy (Step 4).

In the second phase all previously detected interest points are analyzed in order to get their characteristic orientation and corresponding descriptor vector. In order to obtain the dominant orientation, the maximum of an histogram of local gradient angles is determined (Step 5). This orientation assignation is very important, since most of future image operations will be performed on transformed patches relative to this orientation and the natural scale of the feature (obtained in the first phase). Thus, the obtained descriptors will provide the desired invariance to these scale and rotation transformations. The final keypoint descriptor extraction stage is again based on local histograms of orientations, this time on a conveniently scaled and rotated grid of bins (Step 6). Basing

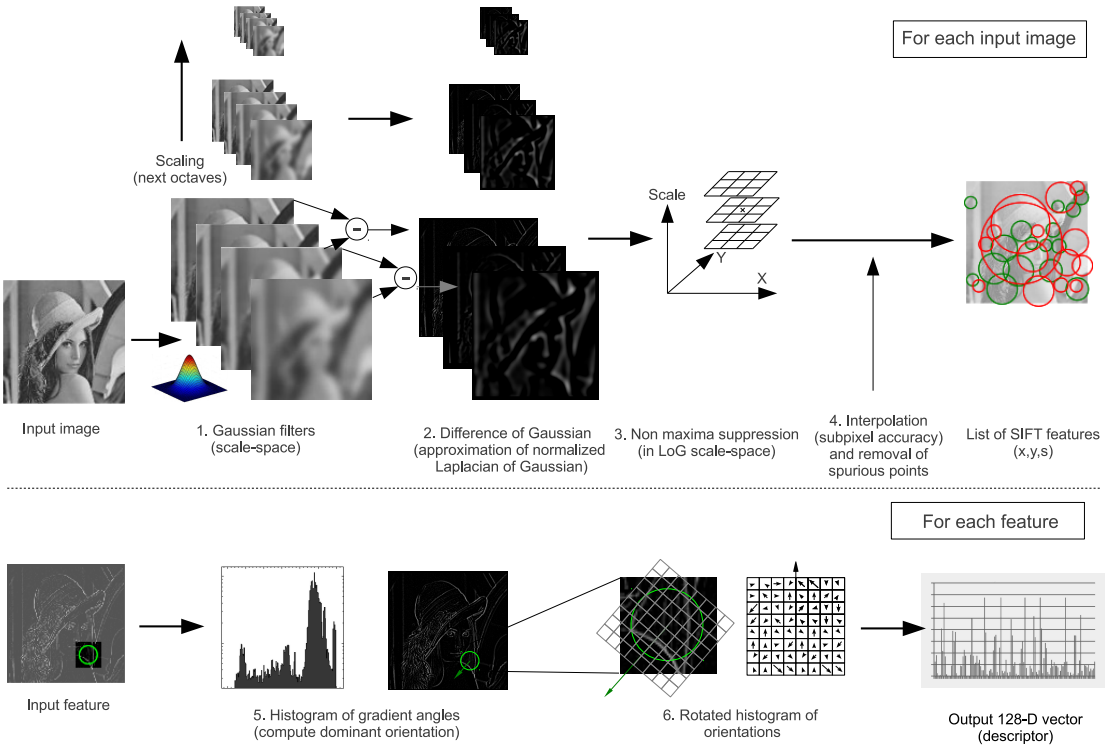


FIGURE 4.5: Main stages of the SIFT procedure.

the descriptor in normalized gradients instead of direct use of gray levels in the image, makes the output 128-D descriptors moderately invariant to shape distortion and very robust to changes in illumination. Though in this work we use the basic SIFT algorithm described above, there are several algorithms to additionally filter the most distinctive feature vectors, such as the ones proposed in [127, 176, 177], which could be considered in future refinements.

Since the SIFT extractor algorithm is based on a computationally intensive pyramid processing of the input image, there are several parameters that strongly influence its performance (mainly the ones related to the overall number of scales, the size of the initial scale and levels of gaussian smoothing by scale). We tested different values for these parameters and selected values were: i) number of scales equal to 5; ii) first scale equal to input image size (no image downscaling or upscaling are initially applied) and iii) 5 levels of gaussian smoothing by scale. These parameters led us to obtain a rough average number of 250 stable keypoints per image on a typical scene at our environment.

The intensive image processing involved by the described algorithm led us to use a GPU based implementation. Though several less computationally intensive alternative feature extractor and descriptor techniques have been described in the literature (see for example [121], and the references therein), they tend to be still too hard to be computed in the smartphones in terms of both processing time and battery consumption.

4.2.3 Location estimation based on SIFT features matching

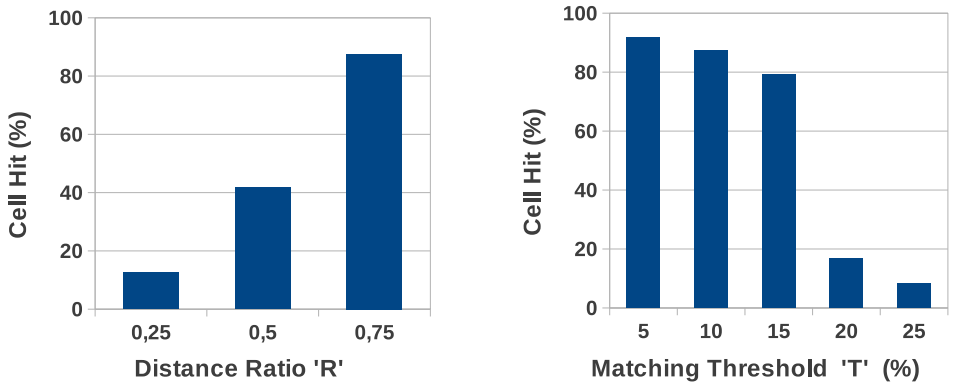
As it has been previously indicated, each keypoint descriptor is 128-dimensional and this implies that the corresponding search space has a very high dimensionality. In consequence, there is no algorithm able to identify the exact nearest neighbors that is any more efficient than exhaustive search. As an alternative, there are some algorithms such as the *best-bin-first* proposed by Beis and Lowe [31] that returns the closest neighbor with high probability. Another option is the proposal made by Arya and Mount [20] based on the use of *kd-trees* and *bd-trees*, which supports both exact and approximate nearest neighbor searches in spaces of multiple dimensions. In our proposal we made use of the kd-tree version implemented within the *approximate nearest neighbor* (ANN) library by Mount and Arya [135] to carry out our experiments.

Our database of image features contains more than 45K data points in real 128-dimensional space, which is a moderately large amount of data. However, if we precompute one of the aforementioned tree structures for these contextual data, the nearest neighbor for new input features can still be found efficiently. Therefore we built a *kd-tree* made up of a complete set of features available in our database. Each feature was labeled with its corresponding cell in order to be able to estimate the current position of a device during the on-line phase.

ANN allows the selection of the number of returned k-nearest neighbors in the solution, where $k \geq 1$. Working with such high dimensional features, global differences of the euclidean distance between descriptors are not useful to differentiate when we have obtained a correct matching, as some descriptors are far more discriminative than others. Thus we need to estimate the density of matchings near the input feature in a different way. Using $k = 2$ we can compute the distance ratio between the two nearest neighbors in order to check whether it is a real match or not. We based our technique on the reasoning made by D. Lowe in [127], who demonstrated that correct matches need to have the nearest neighbor significantly closer than the second neighbor (a supposedly incorrect match) to achieve a reliable matching. As Figure 4.6(a) shows, we evaluated this metric using three different values for this distance ratio R, discarding those matchings with a ratio greater than the specified value.

Tested values were selected according to the information obtained from literature. A priori, we might suppose that using a lower value of R better results would be obtained, since we expected a better reliability of the obtained matchings. However, lower values of R made the process more restrictive and fewer number of matchings were found.

In comparison to the perspective that will be introduced in Section 4.3, where we will analyze the geometric consistency of the obtained matchings, in the current proposal we only consider the global number of matchings obtained from each individual image. Therefore, using restrictive values of R makes it more difficult to find an image that ensures a minimum number of correspondences (fixed to 10% of the total image features to accomplish these experiments). According to the results, the best balance in terms of accuracy and performance has been obtained with $R = 0.75$, reaching an 88% of cell hits (including adjacent cells).



(a) (b)

FIGURE 4.6: Analysis of different input parameter configurations for the matching algorithm; a) Performance evaluation; b) Accuracy evaluation.

Another important parameter of this matching algorithm is the number of individual matches required to assess whether there is an evidence of a global image matching or not. Since the number of features for each image may be so high, we can specify a matching threshold T, which stands for a minimum percentage of matches against the total number of features per image to define a global match as valid. We experimented with different T values (see X axis of Figure 4.6(b)). As we can observe, when T is assigned higher values, accuracy is notably decreased. Once again, relaxed values of T achieve a better performance in terms of accuracy. That is, as T increases we need to obtain a greater number of matchings to consider that an image represents a certain position. The main reason that explains these results is the reduced number of matchings we obtained from tested images, since these experiments were carried out using images collected mainly at corridors, where we noticed an important lack of visual information and textures in walls, doors, etc. In consequence, we finally found that 0.05 is the value that achieves the best accuracy in our scenario. That is, given a query image, we considered that there was a global match when at least 5% of its features were found in a particular image of the database. Note that the selected values may vary depending on the scenario, ensuring an appropriate trade-off between accuracy and performance.

When an approximate nearest neighbor search is performed, an error bound $\epsilon \geq 0$ is needed to control the maximum ratio between the distance to the reported point and the true nearest neighbor, with this ratio equal to $1+\epsilon$. We experimented with different values of ϵ and the results are shown in Table 4.1 using a representative dataset of images

collected at our scenario. From these results we concluded that on enlarging the error bound to $\epsilon = 2$ when using this approximate nearest neighbor search, we speed up the process (taking around 200 ms) without affecting the accuracy. Though using higher values $\epsilon > 2$ we can still accelerate the matching process, it has a negative impact on the quality of the obtained matchings, affecting the final accuracy obtained.

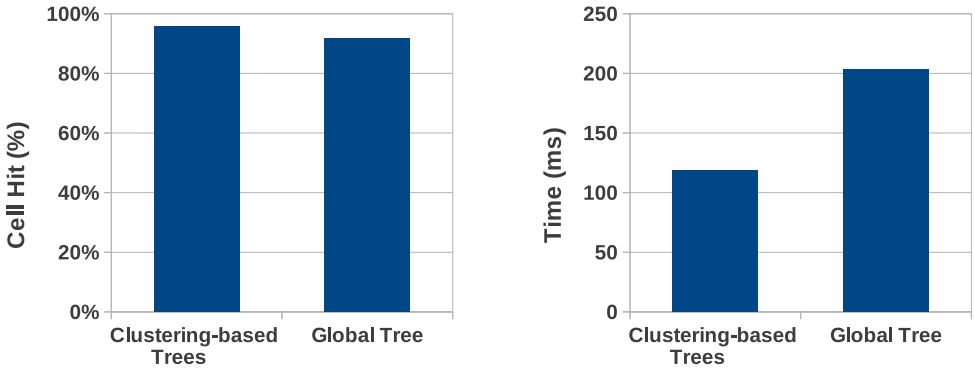
Ɛ	#Matching	Mean Time (ms)	Cell Hit (%)
1	6,6	3128,4	88
2	6,7	207,7	88
4	8,9	81,6	77
8	11,4	49,8	64

TABLE 4.1: Epsilon analysis results. Time data are expressed in milliseconds while accuracy is specified in terms of cell hit (including adjacent cells) percentage.

4.2.4 Multisensor integration

Once the right parameters for an optimal image search have been selected, in this section we focus on how we took advantage of the multisensor integration to improve the performance of the location estimation process without affecting the final accuracy. As mentioned, the main drawback of using images is the elevated computational cost of their analysis, especially in huge scenarios where the number of images is vast, which might imply serious scalability problems. As mentioned in Chapter 3, in [119] Lemelson et al. proposed the Fingerprint Clustering algorithm, which made use of the training WiFi RSSIs to find clusters where signals have similar behavior. We demonstrated that is was possible to obtain a 93% of cluster-hit by applying the appropriate fingerprinting technique. For our specific scenario, we obtained the clusters indicated in Figure 4.2(b). The idea behind the integration of RSSIs and images was that from this clusters distribution it was possible to define several overlapping zones joining adjacent clusters, which allows us to reduce the space search during the matching process, thus improving its performance.

Figure 4.7 shows a comparison of the performance and accuracy results obtained when integrating the division by zones with respect to the use of a unique global tree. In this case we built five different trees, four of them containing the descriptors of those images belonging to each zone, and one additional global tree containing all of them. From the obtained results we can conclude that when using smaller trees we even improve



(a) (b)

FIGURE 4.7: Comparison in terms of (a) accuracy and (b) performance between using clustering-based tress or a unique global tree at matching process.

the system accuracy (around a 5% of cell hit), but what is more important, we also get better performance results, with more than 30% reduction in the search time.

Taking into account that our experimental scenario is comparatively small, we expect that in larger maps this difference between using a global tree or using clustering-based trees will lead to a still higher performance improvement, thus demonstrating the importance of this multisensor proposal to favor system scalability.

In addition, we introduce a method that analyses the suitability of the data acquired. First, information obtained from access points is analyzed in order to determine whether we are in a location supported by our system. Moreover, it is checked whether the image is useful, that is, it is focused and rich enough to allow the extraction of useful SIFT features. In order to avoid unnecessary transmissions, we used the Sobel operator [68], an efficient linear filter operation which responds to sharp gray level changes (edges) in images. When a picture is taken we compute its gradient to know how blurry the image is. If a good image is obtained (gradient value is up to a predefined threshold G), it will be transmitted to be used for localization purposes.

Sensor fusion process

Now we describe the global process carried out in order to fuse the information obtained from the WiFi and camera sensors that allowed us to obtain the results shown in Figure 4.7. Initially, RSSIs are processed to get a probability distribution vector indicating the

likelihood of being located at each cell of the scenario. For that purpose we make use of the fingerprinting-based technique described in Section 3.4 of Chapter 3. Then, SIFT features are computed from the obtained image. Next, using the ANN kd-tree matching algorithm a *pseudo-probability* distribution is obtained according to the number of matches obtained. The matching process continues as follows: We select the subtree linked to the cluster containing the cell with higher probability after the RSSI analysis. Later, we compare the descriptors extracted from the input image with those stored in the selected subtree to look for correspondences. If the matching result does not exceed the threshold T (mentioned in Section 4.2.3), we repeat the same process, but this time selecting another subtree linked to another cluster containing that cell, if available. Finally, if we do not exceed T we select the global tree to perform a final search.

To accomplish this fusion process we made use of the Baye's Rule in a similar way to that used in Section 3.4. Following this approach we obtain a probability distribution indicating the likelihood of being located at each cell of the scenario. Due to the low number of APs deployed in our testbed, we obviated the term that considers the frequency of appearance for APs that was present in the original equation . We considered π as the probability distribution vector over each cell in the previous timestamp, being $C = \{c_1, .., c_m\}$ the set of cells that make up the finite space state. Given the amount of n RSSI measurements in the current observation O_j, and $Pr(\lambda_\beta | a_\beta, c_i)$ the probability of taking a measurement from the access point a_β at reference cell c_i with a signal strength λ_β, there is a first estimation π' based only on RSSI:

$$\pi'_i = \frac{\pi_i Pr(O_j|c_i)}{\sum_{\alpha=1}^{m}(\pi_\alpha Pr(O_j|c_\alpha))} \quad \text{where} \quad Pr(O_j|c_i) = \prod_{\beta=1}^{n} Pr(\lambda_\beta|a_\beta, c_i) \tag{4.1}$$

Considering π'_h as the highest value in π', we can constrain the analysis of the image in Oj to the subtree that represents the zone where c_h is included.

We define $Pr(f|c_c)$ as the probability of seeing an image f at cell c_c contained in the selected zone. All the images related to that zone are analyzed to determine the image with a higher number of matchings. Taking D as the number of SIFT features detected in image f, if there is any cell c_c where $\frac{matches_{f,c_c}}{D} > T$ (T was defined in Section 4.2.3), then $Pr(f|c_c) = K$ being K a predefined constant value (in our case 0.95). We assign this high probability value since we believe there is strong evidence (but not total

certainty) to be located at cell c_c. The remaining probability is spread among the other cells of the entire scenario.

Otherwise, in case of existing other subtrees representing a zone which was also associated with c_h, the same process is repeated using each one of them respectively. If the threshold T is not exceeded we make use of the global tree. Even so, in case of no success, $Pr(f|c_c)$ is defined as follows:

$$Pr(f|c_c) = \frac{matches_{f,c_c}}{\sum_{k=1}^{l}(matches_{f,c_k})} \tag{4.2}$$

Where l represents the total number of images contained in the global tree. Those cells where no matchings are found are assigned a negligible probability value (to avoid zero probability distribution). Finally we recalculate the final probability distribution π'' by fusing the already estimated π' with the corresponding $Pr(f|c)$ probability calculated for each individual cell at the scenario, normalizing the obtained probability vector to ensure a true probability distribution.

$$\pi_i'' = \frac{\pi_i' Pr(f|c_i)}{\sum_{\beta=1}^{m}(\pi_\beta' Pr(f|c_\beta))} \quad \forall i = 1, .., m \tag{4.3}$$

Finally, the cell assigned to the best probability is considered as the estimated position, though the currently obtained probability distribution vector will be used as an input for future estimations. It is worth mentioning that during our experiments more than 83% of estimations were successfully resolved using the subtrees, without requiring the analysis of the global tree.

4.2.5 Accelerometer integration for still/motion state detection

We suggest the integration of a third additional sensor, the accelerometer, in order to increase the robustness of the location estimations based on RSSIs. In addition, this sensor will help us to minimize the amount of information transmitted by the client, preserving the smartphone battery life, which is another keystone within localization systems, and which has been already addressed in works like Sourav [33].

We made use of the built-in 3-axis accelerometer readings to detect changes in motion status. Taking into account the fact that smartphone's accelerometers are usually very noisy, we decided to identify only two different states: *still* and *motion*. These states can be viewed as an approximation of the user motion status. We implemented a simple classifier based on the estimation of a threshold able to differentiate between these two states. In order to train the classifier, we got several sets of samples from the acceleration measurements *(in* m/s^2) of the accelerometer (sampled at 50Hz) using different patterns of movement. To obtain each sample we follow a sliding window approach, considering the average and standard deviation values of the last 10 acceleration readings. The threshold can be defined as the measurement variance that maximized the separation between classes. Figure 4.8 clearly shows the results achieved in the differentiation between classes.

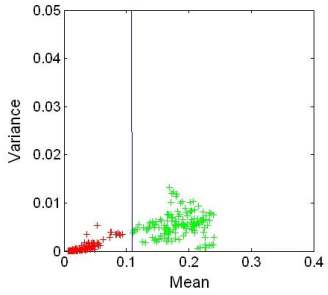


FIGURE 4.8: Motion classification results (red for still, green for motion)

Thought it is not strictly necessary, it is possible to introduce the information obtained by the accelerometer in the fusion process described in Section 4.2.4. As we indicated in Section 3.4 of Chapter 3, we had the possibility of using a HMM as a system model that takes into account the user's motion behavior. Integrating the distinction between *still* and *motion* stages, we can define two different transition matrices. One of them might represent a more static behavior whereas the other one might favor the transition between cells. We will evaluate the implications of using the accelerometer within the location estimation process.

As far as the client is concerned, we can detect those cases in which it would be unnecessary to request continuous estimations about its location. The information obtained by the accelerometer might be used to determine whether the device moves or remains still. When still, once the right position is determined there is no need to obtain a new location estimation. Therefore, regarding the localization server, the integration of this sensor improves its scalability, as it supports a higher number of users due to

the fewer number of queries to be answered. Additionally, from the perspective of the smartphone, the use of the accelerometer implies direct energy saving, since it avoids unnecessary operations and transmissions.

4.2.6 Evaluation

In order to validate the accuracy and performance of this initial multisensor proposal, this section describes the results obtained after performing several experiments. To accomplish them, we made use of different mobile devices running an augmented reality prototype able to display location-aware notes linked to each cell.

During the tests we obtained information from all the available sensors (images, RSSI and accelerometer). We divided the tests into two different categories: still tests where we remained still at the same place, and motion tests where we moved along the available dependencies. The still tests took place at several cells, in corridors, offices and laboratories. During the motion tests, we covered several paths mainly along the corridors. Locations were estimated with three different combinations of sensors: using RSSI only, using RSSI and accelerometer measures, and using all the sensors, in order to check the accuracy of each combination. Table 4.2 shows the results obtained, reflecting the cell hit percentage and in parenthesis the cell hit percentage including adjacent cells as good estimations.

As we can observe, the use of the accelerometer results in an important improvement over the accuracy obtained, especially in the case of the still test where the exclusively RSSI-based estimations are more precise. It allows us to integrate a HMM able to adapt the estimation algorithm according to the motion pattern. In still cases, HMM increases the probability of remaining at same place, whereas in motion cases it favors the movement avoiding the possibility of obtaining inconsistent consecutive estimations. However, the inclusion of images in the positioning algorithm notably boosts the system accuracy, achieving cells hits of more than 97% and 94% (including adjacent) for stationary and movement situations, respectively.

Regarding the system performance, Table 4.3 shows the average time required by the tasks performed by the server in order to reply to a localization request. It includes the time to carry out the SIFT extraction that is computed using the GPU installed on the

Motion Status	Still	Motion
Fused Sensors		
RSSI	20.9 (63.5)	4.1 (22.4)
RSSI + Acc	36.3 (83.5)	6.2 (32.6)
RSSI + Acc + Images	82.8 (97.1)	55.1 (94.4)

TABLE 4.2: Cell hit obtained during our experiments. We show in parenthesis the cell hit percentage including those cases in which the estimation corresponds with a cell adjacent to the correct one.

server. It also includes the time needed for the matching process using a unique global tree or using clustering-based trees. We can confirm the importance of the multisensor integration, which saves around 75% of matching time on average. The time required for carrying out the entire process is around 249 ms, making use of our multisensor proposal.

Action	Time (ms)
SIFT Extraction	111
Matching Global Tree	605
Matching Clustering-based Trees	131
Fusion Process	7
Total Localization *(Using clustering)*	**249**

TABLE 4.3: Performance analysis of the different processes carried out by the localization server (Time in milliseconds).

Table 4.4 shows the average time distribution of the different tasks performed during the whole cycle from the point of view of the clients activity: Column 1 indicates the time required to collect the RSSIs; Column 2 shows the time required for image capturing; Column 3 represents the delay of executing the Sobel operator; and Column 4 shows the time required for data transmission (RSSI, motion estimation and image). We omit the time required to read accelerometer measures, to estimate motion status and to process the captured RSSIs, because of their insignificance. It is important to mention that we applied a compressing ratio to shrink the image file size and reduce the transmission time. This compression did not affect the SIFT features extraction and thus it did not affect the final accuracy.

Though the time required to complete an entire cycle includes the time required to scan every sensor, to process their information and to send it to the server, we can always take advantage of the event-based sensor scanning at Android OS. This means that the data can be acquired and treated in parallel. For example, in the case of the Samsung Galaxy SCL the total time would be 1550 ms, that is, the time capturing the RSSIs (images and accelerometer measures are acquired and treated in parallel) plus the time required for transmission.

Action	RSSI Capture	Image Capture	Sobel Operator	Transmission
Device				
Samsung Galaxy SII	6374	43	34	97
Samsung Galaxy SCL	1476	58	55	74
Samsung Galaxy Tab 7 Plus	2687	62	51	112
Asus EEEPad Transformer	1932	81	60	105

TABLE 4.4: Comparison, using several devices, of the required time for data acquisition (time for accelerometer acquisition is negligible), processing time of the Sobel operator (time for RSSI processing and motion estimation is insignificant) and transmissions. Time is expressed in milliseconds.

As we show, there are important differences among the different devices in relation to the WiFi scanning, depending on the OS version in combination with the hardware components they are equipped with. Manufacturers are highly influenced by current development trends focused on battery consumption saving for mobile devices, and in some cases it constitutes an important drawback for our interests. In this sense, current OS versions of these devices do not support active scans. Consequently we are forced to perform passive scans, which can take a lot of time with some devices to scan each available channel or to perform several scans on the same channel before making the new measures available for their use.

This behavior produces an important delay that might be significantly reduced using future smartphones and OS versions. However, taking advantage of the possibility of capturing measurements from different sensors in a parallel way, we suggest not to wait for a new RSSI scanning to request for a new location estimation. That is, as soon as a new image is available, using the last RSSIs collected, a new location can be estimated. The error introduced by using this cached information is not important since these RSSIs are only used to enclose the tentative zone for image matching. Considering the specific case of the Samsung SCL, the total time required to complete an entire cycle is around 1799 ms (client + server processing time) for the first estimation, though this time can be reduced to 436 millisecond in the following estimations. This delay would include the time required to capture the image, the time to calculate the Sobel operator, the time to transmit the sensors data and the server processing time (accelerometer can be captured and processed in parallel, and we can also cache RSSIs information as stated above). In the particular case of the Samsung Galaxy SII smartphone, the differences are even more noticeable, since we can reduce the response time from 6720 to just 423 milliseconds on average, minimizing the important issue of the delay introduced by the RSSI acquisition.

4.2.7 Summary of the place recognition based proposal

Until now we have described a first approximation to the integration of images for localization purposes. Despite the use of other additional sensors, images have been used as the keystone for this approach, which allowed us to perform an initial analysis of the power of visual information for place recognition based solutions.

The required average time for the whole localization process is 500 milliseconds on average for all tested devices, with cell hit rates up to 94% when users are in motion. In view of the experimental results, we conclude that we are able to provide a good solution for those services where a good accuracy is required and the response time is not crucial for their correct functioning. We consider that the extra time required to deal with images is acceptable for applications requiring a better accuracy than the one obtained using only RSSIs. During the experiments accomplished to evaluate the techniques proposed, some parameters and threshold values have been evaluated empirically due to its dependency on the scenario. Therefore, sometimes we preferred not to indicate specific values.

However, we detected some drawbacks that make this proposal problematic from the point of its scalability. In particular, the training phase implies an enormous effort in terms of the operators work and time. On the one hand, a large amount of RSSIs has to be collected to generate useful fingerprinting maps. On the other hand, the collection of images is also tedious, as several images must be captured at each training point. Moreover, both, RSSIs and images need to be manually geo-tagged with the correct cell where they were obtained. This is a major handicap when modeling scenarios larger than the one we used for our experiments. Though we are able to obtain a good cell hit rate, this precision is dependent on the granularity level used to define the discrete scenario: the more granularity, the better is the accuracy obtained, but clearly at the cost of a greater training effort.

Considering all these issues, in the following section we are presenting an evolution of this proposal in which we try to minimize the observed inconveniences. We will continue using images as the main piece of our localization process, but we evolve our system introducing more sophisticated computer vision techniques. The discrete partition of the environment is left out, and will give way to a new training process making use of techniques that allow the reconstruction of 3D models of the environment. In addition to

the reduction of time required to train the system, this change means an important evolution of the localization process, providing more accurate location estimations, around a few centimeters of error, and with the added value of the estimation of the rotation of the device.

4.3 Image analysis based on 3D models

Though the accuracy obtained using the previously described multisensor approach is good enough for some application scenarios, our main goal is the development of a LBS suitable for precise augmented reality services, which require not only very fine-grained position estimations (a few *cm* of maximum error), but also an equally precise estimation of the absolute orientation of the device.

Taking advantage of the remarkable advances in the field of applied projective geometry in the last decade, nowadays it is possible to obtain an accurate 3D model of an arbitrary scene taking only a sufficiently large number of images from different viewpoints. This can be done even in a priori difficult conditions such as completely uncalibrated cases, in which neither the real position nor the internal parameters of the cameras are known in advance. Making use of these 3D models of the scenes, in this section we propose an evolution of the previous solution, which is able to obtain the precise estimation of the full 6 degrees of freedom (dof) –3 for (X, Y, Z) position and 3 additional for the (α, β, γ) Euler angles for the rotation matrix– of the device with respect to some conveniently chosen world reference coordinate system.

Our proposal is based on the use of Structure from Motion (SfM) techniques to run off-line 3D maps reconstructions of the environment using the SIFT features extracted from the training images. Several resection techniques were evaluated to estimate the full 6 *dof*, depending on whether we know some intrinsic camera parameters, like the focal length. As in the previous case, this new approach uses sensor data, like RSSIs and the built-in accelerometer, but also integrates the measurements from the digital compass. This multisensor integration is proposed in order to limit the amount of information to be examined, to simplify the training process and to reduce the computational load of the smartphones.

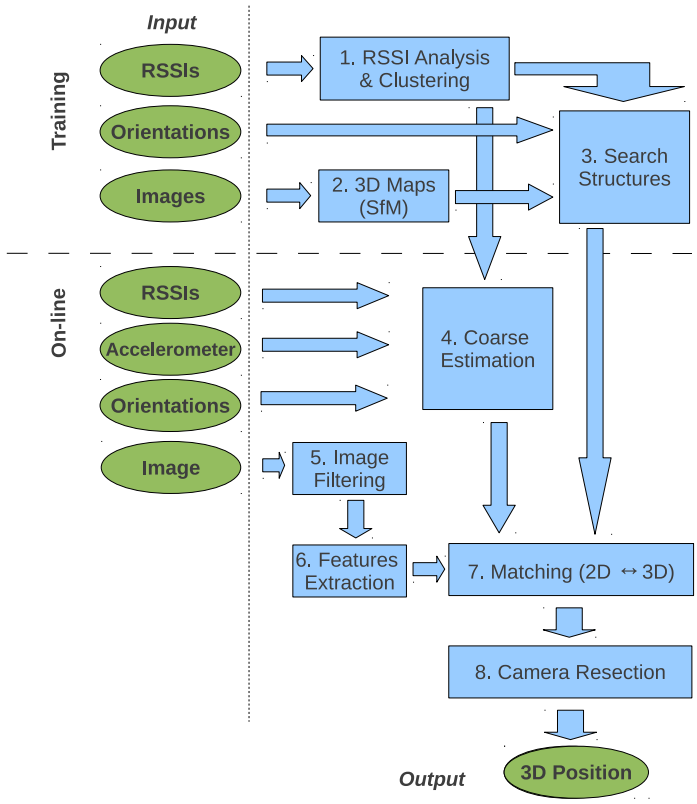


FIGURE 4.9: System overview. It includes all the steps to be taken during training and on-line phases.

For a better comprehension of the rest of the section, Figure 4.9 provides a general overview of the different operations that take place during both the training phase and the on-line phase. During the training phase we noticed some differences with respect to the previous proposal. Now, images, RSSIs and also orientations are captured along the entire scenario. RSSI analysis (Step 1 in Figure 4.9) is performed in the same way as previously commented, in order to build the fingerprinting map and to estimate the clusters. The first important difference takes place at Step 2, where using SIFT features extracted from the images and visual structure from motion techniques we generate the 3D maps of the scene ready to be used for localization. We divide the 3D model according to the geographical areas defined using the clusters distribution obtained from the RSSI analysis, taking also into account the rough orientation of the device (Step 3).

Then, during the on-line phase, multiple sensors are used to determine an initial coarse estimation that indicates a cluster of physical points where the device seems to be, as well as the rough orientation (Step 4). This information will we the key to select the tentative zone for the image analysis. Making use of the image captured by the smartphone, we perform a filtering process based on a gradient model to discard blurry

and uninformative images (Step 5), as described in Section 4.2.4. If the image is valid, we extract its SIFT features (Step 6). Afterwards we perform a matching process, but in this case it is carried out against the features contained in our partitioned 3D model to determine correspondences between image 2D pixels and 3D points (Step 7). Finally we perform a complete camera resection process to estimate the current 3D position of the device (Step 8).

4.3.1 Experimental environment

Once summarized the proposal presented, we describe the experimental environment of our tests. It is a 220 m^2 open space located on the ground floor of the Computer Faculty of the University of Murcia, shown in Figure 4.10. To carry out the experiments, we selected a different scenario from the one previously modeled, motivated by the several difficulties that we found in order to test this new proposal. Firstly, the repeatable nature of our previous scenario (specially at corridors) made unfeasible the reconstruction of a reliable 3D model of the entire scene. Moreover, because of the intrinsic characteristics of SIFT, the lack of texture (also mainly at corridors) prevented us from obtaining a minimum number of features needed for the correct functioning of the SfM techniques.

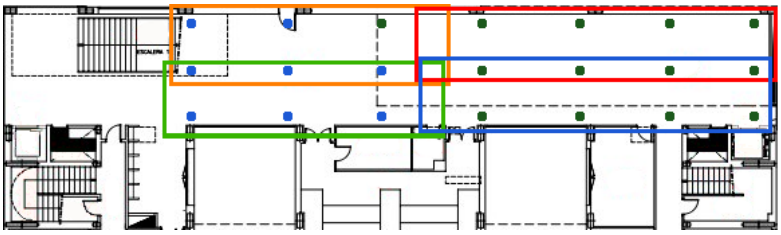


FIGURE 4.10: Experimental environment. Dots represent the centroid of the different cells defined at the discrete space. Colored dots indicate the RSSI-based clusters each cell belongs to. Colored rectangles identify the 3D sub-models defined considering RSSI-based clusters and orientations.

To carry out our experiments we deployed a client-assisted architecture much in the way as the one described in Section 4.2.1. From the client perspective, we used several smartphones and tablets to carry out the training tasks and to test our proposal. More specifically, this new set of experiments were carried out making use of a Samsung Galaxy Tab Plus 7.0 with Android OS 3.2 and an ASUS EEEPad Transformer TF-101 tablet with Android OS 4.0.3.

In this case, our server was hosted in an Intel® Core^{TM} i7-2600K CPU 3.40GHz, using a nVidia GeForce GTX580 with 512 cores, supporting again both GLSL and CUDA. For image processing purposes we made use of the SIFTGPU library [9]. This server was responsible for estimating the location of the different devices. Additionally, we also used the CUDA ENN software developed by V. Garcia [65] to perform the matching process and our own implementation of the resection algorithms making use of the QVision library, available in *SourceForge*. VisualSfM software [11] was used to run the 3D model reconstruction of the testbed scenario.

During the training we recorded a video, and we captured compass measurements and RSSIs observations using a Samsung Galaxy Tab Plus 7.0 with Android OS 3.2, with camera focal $f = 615$ pixels. It is worth mentioning that within this environment we built the fingerprinting map using the RSSIs collected from all the available APs, instead of using a controlled subset of them. One of the main reasons for not performing precise estimations just using RSSIs, was that it was not necessary to have an exhaustive control on the set of used APs. This was not the case in Section 4.2, where we decided to reuse the existing fingerprinting map created for experiments in Chapter 3 to save deployment time.

We used our own application to perform a fast training process. First, we defined a coarse-grained discrete layout to label the sensor measurements geographically. Then, we scanned the RSSIs of the different access points already deployed in the scenario while we were video recording the environment and registering the rough orientation estimated by the rotation sensor of the device. The operator just had to label the current cell where he was moving around. This way, we obtained a lightweight fingerprinting map of RSSIs and a set of images (frames extracted from the recording) geo-tagged with the corresponding cell and orientation. Then, a clustering process was performed using the techniques described in Chapter 3 in order to group these cells into clusters according to the characterization of the RSSIs. Zones were then obtained using the defined clusters and considering to the orientation. Images extracted from the video recording were used to build the 3D model of the scene using the technique introduced in the following section.

Finally, during the on-line phase the system was tested using an ASUS EEEPad Transformer TF-101 tablet with Android 4.0.3, whose camera was precalibrated, with an

estimated focal length of $f = 630$ pixels for our working image size (640×480). We developed an *augmented reality* (AR) application able to collect data from different sensors (WiFi, compass, accelerometer and camera) and to send them to the server, which is in charge of estimating the current position of the device.

4.3.2 Building 3D models of the scene

The myriad of techniques involved in the complex process of 3D models reconstruction, generically known in the computer vision literature as *Structure from Motion* (SfM), goes from feature extraction and matching to estimation of the geometric relationships between the multiple views of the scene, passing through multiple intermediate matrix algebra procedures and both linear and non-linear optimization techniques. Each of these computing stages have generated an enormous amount of research in the last decade, so their thorough descriptions are not within the scope of this dissertation. The remarkable book [77] –the *de-facto bible* in the field– is a good reference for those interested in the underlying details of this research field.

Here we will briefly summarize the overall process followed by a typical SfM system, just in order to get an idea of the general functioning and power of such techniques. Figure 4.11 shows a detailed diagram of the different steps which are performed. We use here, as the illustrating example, the 3D reconstruction of our environment.

During the training phase, a video is recorded capturing all the details of our environment. The video is then processed in order to extract images at a particular frame rate, which will be used as input for the SfM process (Step 0 in Figure 4.11). The use of video recording not only saves a lot of time and effort during training, but also guarantees better 3D reconstructions, since overlapping images and a sequential order facilitate the posterior matching and reconstruction stages. Taking as input only that set of images, and by extracting (Step 1) and matching (Step 2) the SIFT features among them, the aforementioned SfM techniques are able to finely estimate the 3D position of visually relevant points in the scene, as well as the position and orientation of the cameras from where the pictures were taken. In order to do that, the procedure starts from individual stereo pairs (Step 3), which are augmented later with new cameras by an incremental resectioning/triangulation procedure (Step 5). This incremental process is guided by the global set of pairwise matchings matrix found among all the images (Step 4). The

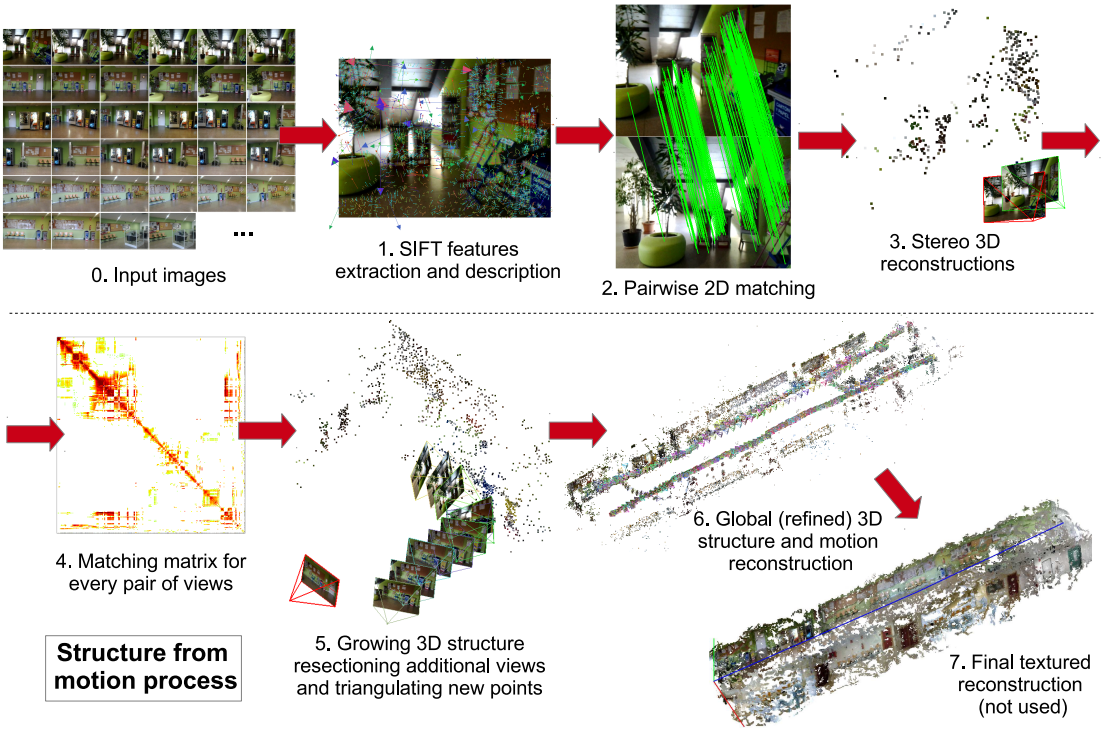


FIGURE 4.11: Illustration of the *SfM* technique. Input images are preprocessed by extracting and matching SIFT features among them, in order to get relevant stereo pairs from which the 3D reconstruction can be bootstrapped. These partial reconstructions are then augmented with new cameras by an incremental resectioning/triangulation procedure. The final step, known as *bundle adjustment* [175], is a global nonlinear optimization which minimizes the reprojection error of all the 3D points to their corresponding 2D image projections. All figures obtained using Changchang Wu's VisualSfM software [11] on our test environment.

final step in every SfM process, known as *bundle adjustment*, is a global nonlinear optimization stage which minimizes the reprojection error of all the reconstructed 3D points to the 2D image position of the original SIFT features in the images, thus producing an accurate 3D reconstruction which includes both the 3D points and the full 6 *dof* position of the cameras (Step 6).

Though not shown in the figure, each resulting 3D point is attached to the set of corresponding 128-dimensional descriptors as extracted from the corresponding images where that point appeared in the field of view, something that will be the key for our posterior localization process. Moreover, though not strictly needed in our system, the set of points can be augmented with texture patches to get a more realistic 3D reconstruction (Step 7). As it can be seen, the obtained reconstructions provide a fairly accurate metric reconstruction of the environment, up to an arbitrary global euclidean transform, which is conveniently defined choosing an adequate world reference coordinate system (red-green-blue axis in the figure).

4.3.3 Image matching against 3D models

Once we have obtained the 3D representation of the scenario, we need a mechanism to look for coincidences between the features extracted from the 2D images captured during the on-line phase and those features that made up each one of the 3D points of the model. In order to perform that matching process efficiently, we considered different search structures that might be suitable to represent the 3D model. In our case, we typically manage sets of several thousands of 3D points which are obtained from hundreds of thousands of features extracted from the training images.

An alternative is to consider greedy algorithms, as proposed in Section 4.2.3. Though useful for dealing with large databases of features (up to millions of them), the moderately large size of a typical individual zone in our case led us to test a brute force alternative. We took advantage of the processing capacities of the GPU available in the server, and we decided to test an alternative based on an *exact nearest neighbor* (ENN) implementation of the brute force search algorithm running on this GPU. To take advantage of the enormous power of these relatively inexpensive computing platforms, we slightly modified the algorithm proposed by V. Garcia in [65] to adapt its functionality to our zone-based space search. Our complete 3D model is made up of $\sim 250K$ descriptors in a real 128-dimensional space, and considering our multisensor integration, we are able to define smaller zones which divide our scenario, what allows us to establish certain restrictions about their size. The use of this algorithm resulted in an increased matching performance of up to 70% in terms of response time with respect to the original ANN implementation. In Section 4.3.5 we will show additional quantitative and qualitative information about the matching process.

4.3.4 Camera resection

Now we provide detailed information about the camera resection techniques that have been evaluated in order to obtain the precise position of the device. Camera *resection* (sometimes generically referred to as camera *calibration*) is the process by which we can determine the $\mathtt{P}_{3\times4}$ *projection matrix* from a given set of matching 3D scene points and the corresponding 2D pixels in an image. This matrix defines the algebraic relationship $\mathtt{P}\boldsymbol{X}_i = \boldsymbol{x}_i$ for every $\boldsymbol{X}_i \leftrightarrow \boldsymbol{x}_i$ correspondence when both the pixels $\boldsymbol{x}_i$ and the 3D points

$\boldsymbol{X}_i$ are given in homogeneous coordinates [77]. $\mathtt{P}$ can be factorized as $\mathtt{P} = \mathtt{K}_{3\times3}\mathtt{R}_{3\times3}[\mathtt{I}|-C]_{3\times4}$, in a way that the internally encoded relationship with the rotation $\mathtt{R}$ and 3D position $C = (c_x, c_y, c_z)^\top$ of the camera in the reference 3D coordinate system is made explicit. This factorization depends also on $\mathtt{K}$, the so-called camera *calibration matrix* defined as $\mathtt{K} = diag(f, f, 1)$, where f is the camera focal in pixels. We provide different solutions in order to solve the problem in *uncalibrated* cases, when f is unknown and must be automatically estimated, but also in *precalibrated* cases, in which f is known in advance. Finally $\mathbf{x}_i = \mathtt{K}^{-1}\mathbf{x}_i^p$ is the way to obtain the working coordinates $\mathbf{x}_i$ from the original, image centered pixel coordinates $\mathbf{x}_i^p$.

As already mentioned, the main goal basically consists on the estimation of $\mathtt{R}$ and C from potential matches $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$. However, the SIFT matching process is always contaminated with a variable proportion ρ of *outliers*, i.e. wrong correspondences [127]. To filter them out, the robust RANSAC algorithm [61] selects random minimal subsets of the original correspondences set, from which it computes tentative locations of the device. In this section we present three different approaches: the *direct linear transformation* (DLT) algorithm [77] for the uncalibrated cases (i.e., unknown $\mathtt{K}$) and the Fiore linear pose estimation algorithm *exterior orientation* [60] and the *perspective 3 points* (P3P) [61] algorithm for the precalibrated case.

DLT and Fiore's algorithms have the advantage of being based on simple linear algebra procedures, but unfortunately need minimal subsets of 6 and 5 matchings, respectively. Since the RANSAC algorithm works iteratively by finding solutions from tentative random subsets of these minimal sizes until it eventually finds a subset free of outliers, the probability of finding a good subset diminishes exponentially with the size of such subsets. Thus, as a refinement of the Fiore's algorithm, we propose a third resection procedure, the P3P algorithm, which is based on an *algebraic minimal solution* which only needs 3 correspondences. As in the case of Fiore's, the P3P algorithm is only applicable in precalibrated cases. Though this algorithm gets slightly more complicated, as it is not linear anymore, the advantages are far greater because the theoretical probability of finding a good minimal solution earlier gets boosted, as we will confirm experimentally in Section 4.3.5.

The overall resection process is detailed in Algorithm 1. It starts by running a robust implementation of the RANSAC process using the DLT algorithm, which allows us to

obtain the camera matrix P that minimizes the reprojection error for the maximum number of *inlier* correspondences. In this context, inliers are matching pairs of 3D↔2D points which can be fitted to the model up to a given error tolerance. Then, and if the camera has been precalibrated, we discard P, using only the obtained inlier set to get the desired camera pose $(\mathtt{R}, C)$ and applying the *Fiore's* algorithm. Otherwise, we still can perform a camera *autocalibration*, using the previously estimated P to solve the factorization problem by carrying out the simple and well known matrix algebra operation of QR decomposition [67]. These alternatives give us the possibility to treat both the cases of knowing the camera internal parameters in advance (feasible for most common devices), as well as the completely uncalibrated case, which is solved at the price of a slightly lesser precision. A final pose is obtained by refining the former solution using some standard non-linear optimization procedures [141].

Algorithm 1 Resection algorithm

Input: $2D \leftrightarrow 3D$ correspondences, optional calibration matrix K_{fixed}

1: $\mathtt{P}, inliers \leftarrow DLT(correspondences)$ ▷ RANSAC
2: **if** camera intrinsic known **then** ▷ Precalibrated case
3: $\quad \mathtt{K}, \mathtt{R}, C \leftarrow Fiore(\mathtt{K}_{\mathtt{fixed}}, inliers)$
4: **else** ▷ Autocalibration
5: $\quad \mathtt{K}, \mathtt{R}, C \leftarrow QR_{Decomposition}(P)$
6: **end if**
7: $\mathtt{R}_{\mathtt{opt}}, C_{opt} \leftarrow NonLinearOptimization(\mathtt{K}, \mathtt{R}, C)$

Output: $\mathtt{R}_{\mathtt{opt}}$ and C_{opt} (camera pose in world coordinates)

Despite the lower number of correspondences required by the RANSAC procedure encapsulating Fiore's algorithm (5 correspondences against the 6 of using DLT), we decided to perform the RANSAC process using DLT with the main aim of implementing a generic solution able to support both non-precalibrated and precalibrated situations. Since the computational overload difference is not excessive, it does not affect the performance.

In the case of $P3P$, the process was addressed in a different way. As fewer inliers are required (only 3), the RANSAC process is not as exhaustive. This technique was introduced as an alternative to Fiore's, and means a leap forward in terms of accuracy and performance for the precalibrated case. Considering the important performance differences between the RANSAC processes in DLT and P3P, we finally provided and tested a separate solution for this case.

4.3.4.1 Camera auto-calibration case

Direct Linear Transformation (DLT) is the solution proposed in those cases in which camera calibration parameters are a-priori unknown. Here the P matrix that we must first estimate is an homogeneous entity with 12 elements, so that it has in general 11 *dof*. Each 3D↔2D correspondence $(x, y, z) \leftrightarrow (p, q)$ sets up the homogeneous restriction $\mathtt{P}(x, y, z, 1)^\top = \lambda(p, q, 1)^\top$, which is equivalent to $(p, q, 1) \times \mathtt{P}(x, y, z, 1) = (0, 0, 0)$. Namely,

$$\begin{bmatrix} \mathbf{0}^\top & -\boldsymbol{X}_i^\top & q_i \boldsymbol{X}_i^\top \\ \boldsymbol{X}_i^\top & \mathbf{0}^\top & -p_i \boldsymbol{X}_i^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{P}^1 \\ \boldsymbol{P}^2 \\ \boldsymbol{P}^3 \end{bmatrix} = \mathbf{0} \tag{4.4}$$

where $\boldsymbol{X}_i = (x_i, y_i, z_i, 1)^\top$ and $\boldsymbol{x}_i = (p_i, q_i, 1)^\top \ \forall i$ are the homogeneous coordinates of the corresponding pair of points, and $\boldsymbol{P}^k = (p_{4k-3}, p_{4k-2}, p_{4k-1}, p_{4k})^\top$ for $k = 1, 2, 3$, are the three rows of P. Taking into account that we must solve for 11 unknown (*dof* of P), and since each $2D \leftrightarrow 3D$ correspondence gives rise to two independent equations over the elements of P, it is necessary to use of a minimum of 6 of these correspondences to solve the complete system. From the solution for the 12 unknown $p_1 \cdots p_{12}$ we can easily reconstruct a canonical version of the homogeneous matrix P:

$$\mathtt{P} = \begin{bmatrix} \frac{p_1}{p_{12}} & \frac{p_2}{p_{12}} & \frac{p_3}{p_{12}} & \frac{p_4}{p_{12}} \\ \frac{p_5}{p_{12}} & \frac{p_6}{p_{12}} & \frac{p_7}{p_{12}} & \frac{p_8}{p_{12}} \\ \frac{p_9}{p_{12}} & \frac{p_{10}}{p_{12}} & \frac{p_{11}}{p_{12}} & 1 \end{bmatrix} \tag{4.5}$$

The former procedure can easily be adapted to the overdetermined case (more than 6 correspondences). Since points are measured inexactly due to noise, some of these correspondences may not be fully compatible with any projective transformation, and therefore the best possible P has to be determined, which is found by minimizing the reprojection error in the minimum square cost sense.

As already mentioned, the SIFT matching process frequently "contaminates" the set of real correspondences with some subset of *outliers* (wrong correspondences), which would seriously distort the obtained solution if not adequately filtered. To solve this

problem, a robust estimation procedure is needed, and in this work we use the well known RANSAC algorithm [61]. This algorithm works by finding the best transformation matrix that minimizes the reprojection error taking into account only inlier (i.e. correct) correspondences. To filter the outliers (incorrect correspondences), RANSAC iteratively selects a random subset of six elements from the original set of correspondences, from which it computes a tentative `P`. Since the number of available correspondences is notably higher than this required minimum of six, at each RANSAC iteration we perform an incremental process that iteratively adds the correspondences identified as inliers and recalculates P until no more inliers are detected. One correspondence is identified as an inlier if its reprojection error does not exceed a specified threshold. This iterative procedure, which tries several tentative `P` matrices before succeeding, finishes when it either obtains a sufficient percentage of inliers over the total number of input correspondences, or when the number of iterations exceeds a given threshold. In Section 4.3.5 we will show the influence of all these parameters over both the accuracy and the performance of the results.

The `K`, `R` and C values can be easily obtained from the QR decomposition of the first three columns of `P`. Therefore, at least theoretically, this generic procedure could be applied without a previous knowledge of the calibration matrix of the camera. However, the main problem with this approach is that the result values that we obtain (mainly `K` and C) can be severely coupled, that is, they are not mutually independent from each other. Particularly, in some ill-conditioned cases such as scenes with a dominant plane (a not so uncommon case in typical indoor scenarios, where large walls can cover most of the image), the autocalibration process can fail to give an accurate solution. As Section 4.3.5 shows, the QR decomposition can still be a good alternative when the camera intrinsic parameters are completely unknown, except under the aforementioned ill-posed conditions. For a simpler explanation, hereinafter we will refer to the entire process which includes *DLT + QR decomposition* as the DLT algorithm. In the following sections we show how this a priori knowledge of calibration largely improves the quality of the results.

4.3.4.2 Camera pre-calibrated case

In many cases, the on-board camera specifications (mainly focal length and aspect ratio) of typical smartphones and tablets are known, so the calibration matrix $\mathtt{K}_{\mathtt{fixed}}$ can be determined. Here we present two different alternatives to perform the camera resection in precalibrated situations.

Fiore's algorithm

In 2001 Paul D. Fiore [60] proposed a linear method to obtain only the exterior orientation of the camera, which using its intrinsic parameters results in a more precise estimation of $\mathtt{R}$ and C. Moreover, it performs remarkably well even in the planar case, poorly conditioned for the DLT algorithm, as commented in the previous subsection. Fiore's algorithm requires as input a set of $2D \leftrightarrow 3D$ correspondences (at least 5) previously calculated by running the RANSAC process. The algorithm works in two stages. First it estimates the unknown depths ζ_i for each homogeneous 2D point $\boldsymbol{x}_i$, which gives an intermediate set of (inhomogeneous) 3D points $\zeta_i {\mathtt{K}_{\mathtt{fixed}}}^{-1} \boldsymbol{x}_i$, and then what is left is an *absolute 3D orientation* problem, whose solution yields the desired values of $\mathtt{R}$ and C.

More specifically, given a number of input 2D↔3D point correspondences, $\boldsymbol{x}_i \leftrightarrow \boldsymbol{X}_i$, and the intrinsic camera matrix $\mathtt{K} = \mathtt{K}_{\mathtt{fixed}}$, we are required to find a rotation $\mathtt{R}$, a vector C (attitude and position of the camera) and a subsidiary depths vector $\boldsymbol{\zeta} = (\zeta_1, \cdots, \zeta_i, \cdots, \zeta_n)^\top$ (where n is the number of correspondences) such that:

$$\zeta_i \boldsymbol{x}_i = \mathtt{KR}[\mathtt{I}|-C]\boldsymbol{X}_i \Leftrightarrow \mathtt{K}^{-1}\zeta_i \boldsymbol{x}_i = \mathtt{R}[\mathtt{I}|-C]\boldsymbol{X}_i \quad \forall i \tag{4.6}$$

In order to solve the unknown values of ζ_i, we first compose the matrix $\mathtt{M}$, whose columns are formed with the homogeneous coordinates of the 3D points,

$$\mathtt{M}_{4\times n} = [\boldsymbol{X}_1 \cdots \boldsymbol{X}_n] \tag{4.7}$$

and another matrix $\mathtt{N}$, arranging in this case the homogeneous coordinates $\boldsymbol{x}_i = (p_i, q_i, 1)^\top$ of the 2D points in the following way:

$$\mathtt{N}_{3n\times n} = \begin{bmatrix} \boldsymbol{x}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{x}_2 & \dots & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{x}_n \end{bmatrix} \tag{4.8}$$

Then, taking the *singular value decomposition* (SVD) [67] of $\mathtt{M}$, we obtain the factorization $\mathtt{M} = \mathtt{U}_{4\times 4}\mathtt{D}_{4\times n}\mathtt{V}^T_{n\times n}$, where $\mathtt{U}$ and $\mathtt{V}$ are orthogonal and $\mathtt{D}$ is diagonal. Being r the rank of $\mathtt{M}$, we denote by $\mathtt{V_2}$ the submatrix of size $n \times (n-r)$ which results from taking only the last $n-r$ columns of $\mathtt{V}$. This submatrix spans the null space of $\mathtt{M}$, so that $\mathtt{MV_2} = \mathbf{0}$.

Given the former definitions, it can be shown [60] that the depths vector $\boldsymbol{\zeta}$ can be recovered linearly (up to a scale factor), just by solving the following null-space problem:

$$((\mathtt{V_2}^\top \otimes \mathtt{K}^{-1})\mathtt{N})\boldsymbol{\zeta} = \mathbf{0} \tag{4.9}$$

Where the $\otimes$ symbol stands for the Kronecker product [88] of matrices. Once the depths have been obtained, our initial problem (Equation 4.6) is now reduced to find the correct alignment of two sets of 3D points, a case of the well known absolute 3D orientation problem.

This last stage of the exterior orientation algorithm proceeds then as follows. Let W_i be the 3D depth recovered points, $W_i = (\zeta_i \cdot (x_i/f), \zeta_i \cdot (y_i/f), \zeta_i)^\top$ (where we have used the $\mathtt{K_{fixed}} = diag(f, f, 1)$ assumption), and Y_i the likewise inhomogeneous original 3D points, $Y_i = (x_i, y_i, z_i)^\top$. Then, given these two sets of 3-D points W_i and Y_i, we are required to find the rotation matrix $\mathtt{R}$, the vector C and the scalar s, such that:

$$W_i = s(\mathtt{R}Y_i + C) \quad \forall i = 1...n \tag{4.10}$$

Now, centering the point clouds on their respective means, i.e., $\overline{W}_i = W_i - \frac{1}{n}\sum_{j=1}^{n} W_j$ and $\overline{Y}_i = Y_i - \frac{1}{n}\sum_{j=1}^{n} Y_j$, the scale s is very easy to obtain as $s = \frac{||\overline{W}_i||}{||\overline{Y}_i||}$, for every i. In practice, s can be averaged for the set of points $\forall i = 1 \dots n$, and then used to scale according to the values of $\overline{Y}_i$.

With the sets of points adequately scaled and centered, we are left with the problem of estimating the unknown rotation between $\overline{W}_i$ and $s\overline{Y}_i$. This is known as the *orthogonal Procrustes* problem [70]. Being $\mathtt{W}_{3\times n}$ and $\mathtt{Y}_{3\times n}$ the matrices formed by stacking the points $\overline{W}^i$ and $s\overline{Y}^i$ respectively, the solution is found using again the SVD decomposition. The sought rotation is given by:

$$\mathtt{R} = \mathtt{V}' \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathrm{D}et(\mathtt{U}'\mathtt{V}'^{\top}) \end{bmatrix} \mathtt{U}'^{\top} \tag{4.11}$$

where $\mathtt{U}'_{3\times 3}\mathtt{D}'_{3\times 3}\mathtt{V}'_{3\times 3}{}^{\top}$ stands now for the SVD of the matrix $\mathtt{Y}\mathtt{W}^{\top}$, and the determinant of $\mathtt{U}'\mathtt{V}'^{\top}$ is always 1 or -1.

Finally, once we have calculated the rotation matrix $\mathtt{R}$, we can obtain the translation vector C from (4.10) simply this way:

$$C = \frac{1}{s}(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{W}_i) - \mathtt{R}(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{Y}_i) \tag{4.12}$$

As we will demonstrate in Section 4.3.5, this precalibrated exterior orientation procedure ensures both robustness and accuracy in the estimated camera pose, even in environments where we have to deal with essentially planar scenes.

Perspective 3-Points (P3P) algorithm

In this section we propose an alternative technique to estimate the camera 6 dof in precalibrated cases. In this case the resection procedure is based on an *algebraic minimal solution* which only needs 3 correspondences to estimate the 3D camera center and rotation. This reduced number of correspondences (with respect to the 6 previously used) will lead to a noticeable boost in the resection process. Moreover, as it will be demonstrated, accuracy is also improved using this solution. The so-called *P3P algorithm* is described in-depth in [61], nevertheless we summarize its operation as adapted to our system in Algorithm 2.

Algorithm 2 : P3P algorithm

Input: $\{(x_a, y_a), (x_b, y_b), (x_c, y_c)\} \leftrightarrow \{(X_a, Y_a, Z_a), (X_b, Y_b, Z_b), (Z_c, Y_c, Z_c)\}$

$R_{ij} \leftarrow \|(X_i - X_j, Y_i - Y_j, Z_i - Z_j)\|_2, \quad \forall i, j \in \{a, b, c\}, i \neq j$
$\cos(\theta_{ij}) \leftarrow ((x_i, y_i, 1) \cdot (x_j, y_j, 1))/(\|(x_i, y_i, 1)\|_2 \, \|(x_j, y_j, 1)\|_2), \quad \forall i, j \in \{a, b, c\}, i \neq j$

$K_1 \leftarrow R_{bc}^2/R_{ac}^2$;
$K_2 \leftarrow R_{bc}^2/R_{ab}^2$;
$K_3 \leftarrow (K_1 K_2 + K_1 - K_2)$;
$K_4 \leftarrow (K_1 K_2 - K_1 + K_2)$;
$K_5 \leftarrow (K_1 K_2 - K_1 - K_2)$;

$G_0 \leftarrow K_3^2 - 4K_1^2 K_2 \cos(\theta_{ac})^2$;
$G_1 \leftarrow 4K_3 K_2 (1 - K_1) \cos(\theta_{ab}) + 4K_1 (K_4 \cos(\theta_{ac}) \cos(\theta_{bc}) + 2K_1 K_2 \cos(\theta_{ab}) \cos(\theta_{ac})^2)$;
$G_2 \leftarrow (2K_2(1 - K_1) \cos(\theta_{ab}))^2 + 2K_3 K_5 + 4K_1((K_1 - K_2) \cos(\theta_{bc})^2 +$
$(1 - K_2) K_1 \cos(\theta_{ac})^2 - 2K_2(1 + K_1) \cos(\theta_{ab}) \cos(\theta_{ac}) \cos(\theta_{bc}))$;
$G_3 \leftarrow 4K_5 K_2 (1 - K_1) \cos(\theta_{ab}) + 4K_1 \cos(\theta_{bc})(K_4 \cos(\theta_{ac}) + 2K_2 \cos(\theta_{ab}) \cos(\theta_{bc}))$;
$G_4 \leftarrow K_5^2 - 4K_1 K_2 \cos(\theta_{bc})^2$;

for all x such that $\sum_{k=0}^{4} G_k x^k = 0$ **do** ▷ Solve 4^{th} degree polynomial, up to 4 solutions.
$m^* \leftarrow 1 - K_1$;
$p^* \leftarrow 2(K_1 \cos(\theta_{ac}) - x \cos(\theta_{bc}))$;
$q \leftarrow x^2 - K_1$;

$m'^* \leftarrow 1$;
$p'^* \leftarrow -2x \cos(\theta_{bc})$;
$q' \leftarrow x^2(1 - K_2) + 2xK_2 \cos(\theta_{ab}) - K_2$;

$d_a \leftarrow R_{ab}/\sqrt{(x^2 - 2x \cos(\theta_{ab}) + 1)}$;
$d_b \leftarrow d_a x$;

if $m'^* q \neq m^* q'$ **then**
$d_c \leftarrow d_a(p'^* q - p^* q')/(m^* q' - m'^* q)$;
else
$d_c \leftarrow d_a \left(\cos(\theta_{ac}) \pm \sqrt{\cos(\theta_{ac})^2 + (R_{ac}^2 - d_a^2)/d_a^2} \right)$;
end if
Obtain $\mathtt{R}$ and C from depths d_a, d_b, d_c solving the *exterior orientation problem.*
end for

Output: Up to 4 solutions for absolute orientation $\{\mathtt{R}_l, C_l\}$, $l = 1 \ldots 4$.

Once again, the algorithm first finds the depths (d_a, d_b, d_c) of the $\mathbf{X}_{i=a,b,c}$ 3D points, i.e., their euclidean distance to camera center C. But this time the linear procedure used by Fiore's algorithm is not possible, and the reduced number of matchings forces us to solve a 4^{th} degree polynomial. Thus, in fact this method can produce up to 4 valid distinct solutions (d_a^l, d_b^l, d_c^l), $l = 1 \ldots 4$. Once depths have been obtained, the estimation of the position C and rotation R of the device reduces to find the correct alignment of two triplets of 3D points, i.e. $(X_i, Y_i, Z_i) \leftrightarrow d_i \frac{(x_i, y_i, 1)}{\|(x_i, y_i, 1)\|_2}$ for $i = a, b, c$. This is equivalent to solve the exterior orientation problem, which is done in exactly the same way described in the previous section for Fiores's algorithm.

Tentative solutions produced by successive executions of the P3P algorithm are then tested against the whole set of matches, measuring the *reprojection errors* ξ_i of $\mathtt{P}\mathbf{X}_i$ with respect to the corresponding $\mathbf{x}_i$. A given match $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ is considered an inlier if such ξ_i is below a given threshold ϵ. Once the number of inliers rises the expected proportion $1-\rho$, the RANSAC iteration stops, as we find a valid solution confirmed by a sufficient number of matchings. If, on the contrary, the number of P3P attempts exceeds a given maximum without having found a valid solution, the resection is considered unsuccessful, as it can't return a reliable device position.

4.3.4.3 Nonlinear optimization

As indicated in Algorithm 1, a nonlinear optimization is tipically carried out as the last step of the resection process. This last optimization stage can be used to refine the solutions obtained by any of the different techniques presented. This is very convenient in our case for two main reasons: on the one hand, the linearity of the DLT and Fiore's algorithms implies that they try to minimize an *algebraic* instead of a *geometric* error [77], which can bias the obtained solutions for the $(\mathtt{K}, \mathtt{R}, C)$ to suboptimal results, even if we are using a fairly large number of inliers; on the other hand, in the P3P case, which is in principle limited to get the solution for three matchings, a final nonlinear polishing of the solution is also needed in order to minimize the geometric reprojection error in the complete final set of obtained inliers.

Several alternatives are also available for this task. One simple approach is to use a generic unconstrained search optimization algorithm, which only needs a way to evaluate the cost in each point close to the solution, without needing further analytical knowledge of the cost function. This was our first alternative when testing DLT and Fiore's algorithms, and results shown in Section 4.3.5 were obtained making use of a nonlinear optimization technique of this kind provided by the GNU Scientific Library. Anyway, our analytical knowledge of the reprojection cost error function also allows us to use a more specific technique, namely the Gauss-Newton algorithm [141], which was easily adapted to be used in our case. This refined optimization technique was tested together with the P3P algorithm, following the evolution of our developments, and was finally found to obtain the best results.

Though the reader is referred to [141] for further details on the Gauss-Newton technique, its algorithmic version for our specific resection problem is included in Algorithm 3 for completeness. We also describe here briefly its overall operation. Once the set of inliers is identified, the final pose is obtained by refining the P3P solution using all the available valid correspondences, starting from the initial values of $\mathtt{R}$ and C. Our nonlinear Gauss-Newton optimization algorithm implementation iteratively minimizes the sum of all *alignment errors* between the unit vectors $\frac{(x_i,y_i,1)}{\|(x_i,y_i,1)\|_2}$ and the corresponding $\frac{X_i-C}{\|X_i-C\|_2}$ for all valid correspondences i. This minimization is based on solving a linear equation whose coefficient matrix depends on the derivatives of the individual components of this cost with respect to the six parameters (α, β, γ) of $\mathtt{R}$ and (c_x, c_y, c_z) of C. The process ends when a situation of convergence is reached or a maximum number of iterations (max_{iters} variable in Algorithm 3) is exceeded. This procedure can be optionally made incremental by adding new correspondences identified as inliers according to the current $\{\tilde{\mathtt{R}}, \tilde{C}\}$ until no more inliers are found.

4.3.5 Evaluation

Once we have introduced each one of the different techniques involved in our solution, we present some experimental results in order to validate the proposal. Firstly we provide additional implementation details about the 3D map reconstruction. Afterwards we present the evaluation results obtained after testing the different resection techniques which have been described throughout this chapter.

Building 3D maps

Our 3D model was built using the VisualSFM software [11], a visual structure from motion application that makes use of a set of input images to run 3D reconstructions. VisualSFM relies on the SIFTGPU library [9] to extract the SIFT features and to perform the pairwise image matching process, which must be supervised by an operator in order to avoid erroneous correspondences among features obtained from similar objects which appear in different parts of the scenario and thus are not actually the same. The bundle adjustment is implemented using a GPU version that exploits the hardware parallelism. The fact of requiring the supervision of an operator to build feasible 3D

Algorithm 3 : Nonlinear Gauss-Newton refinement of $\mathtt{R}$ and C

Input: $\mathtt{R}$, C and inlier set $\{\mathbf{X}_i = (X_i, Y_i, Z_i, 1) \leftrightarrow \mathbf{x}_i = (x_i, y_i, 1)\}_{i=1}^{n}$ estimated by P3P.

$\tilde{C} = (\tilde{c}_x, \tilde{c}_y, \tilde{c}_z) \leftarrow C = (c_x, c_y, c_z)$;

$\mathtt{R} \leftarrow \tilde{\mathtt{R}}$;

$\hat{\mathbf{x}}_i = (\hat{x}_i, \hat{y}_i, 1) \leftarrow \mathtt{R}^\top \mathbf{x}_i$; $\quad \forall i = 1..n$;

while not convergence of $\{\tilde{\mathtt{R}}, \tilde{C}\}$ **and** $N_{iters} < max_{iters}$ **do**

$\delta X_i \leftarrow X_i - \tilde{c}_x$;

$\delta Y_i \leftarrow Y_i - \tilde{c}_y$;

$\delta Z_i \leftarrow Z_i - \tilde{c}_z$; $\quad \forall i = 1..n$

$l_i \leftarrow \sqrt{\delta X_i^2 + \delta Y_i^2 + \delta Z_i^2}$;

$l'_i \leftarrow \sqrt{\hat{x}_i^2 + \hat{y}_i^2 + 1}$; $\quad \forall i = 1..n$

$$J_i \leftarrow \begin{bmatrix} \frac{\delta Y_i}{l_i} & \frac{\delta Z_i}{l_i} & 0 & \frac{-(\delta Y_i)^2-(\delta Z_i)^2}{l_i^3} & \frac{(\delta X_i)(\delta Y_i)}{l_i^3} & \frac{(\delta X_i)(\delta Z_i)}{l_i^3} \\ \frac{-\delta X_i}{l_i} & 0 & \frac{\delta Z_i}{l_i} & \frac{(\delta X_i)(\delta Y_i)}{l_i^3} & \frac{-(\delta X_i)^2-(\delta Z_i)^2}{l_i^3} & \frac{(\delta Y_i)(\delta Z_i)}{l_i^3} \\ 0 & \frac{-\delta X_i}{l_i} & \frac{-\delta Y_i}{l_i} & \frac{(\delta X_i)(\delta Z_i)}{l_i^3} & \frac{(\delta Y_i)(\delta Z_i)}{l_i^3} & \frac{-(\delta X_i)^2-(\delta Y_i)^2}{l_i^3} \end{bmatrix} \quad \forall i = 1..n$$

$r_i \leftarrow (\delta X_i/l_i - \hat{x}_i/l'_i, \delta Y_i/l_i - \hat{y}_i/l'_i, \delta Z_i/l_i - 1/l'_i)^\top$;

Stack J_i matrices and r_i vectors $\forall i$ to form matrix $J_{3n\times 6}$ and vector $r_{3n\times 1}$.

$(\Delta\alpha, \Delta\beta, \Delta\gamma, \Delta c_x, \Delta c_y, \Delta c_z) \leftarrow (J^\top J)^{-1} J^\top \cdot r$;

$$\tilde{\mathtt{R}} \leftarrow \tilde{\mathtt{R}} \begin{bmatrix} \cos(\Delta\alpha) & \sin(\Delta\alpha) & 0 \\ -\sin(\Delta\alpha) & \cos(\Delta\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\Delta\beta) & 0 & \sin(\Delta\beta) \\ 0 & 1 & 0 \\ -\sin(\Delta\beta) & 0 & \cos(\Delta\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\gamma) & \sin(\Delta\gamma) \\ 0 & -\sin(\Delta\gamma) & \cos(\Delta\gamma) \end{bmatrix}$$

$\tilde{C} \leftarrow \tilde{C} + (\Delta c_x, \Delta c_y, \Delta c_z)$;

$\hat{\mathbf{x}}_i = (\hat{x}_i, \hat{y}_i, 1) \leftarrow \tilde{\mathtt{R}}^\top \mathbf{x}_i, \forall i = 1..n$;

end while

Output: Final polished solution $\{\tilde{\mathtt{R}}, \tilde{C}\}$

reconstructions is an important issue that will be addressed in Chapter 5, where we will propose a solution to fully automate this process.

In addition to the 3D model, VisualSFM also generates an output file containing the technical information about the model. This information includes the camera parameters, such as focal length (which can be automatically estimated or manually established previous to the reconstruction), quaternion rotation, camera center and radial distortion, estimated for each one of the images (considered as individual cameras) included in the reconstruction. Moreover, it also contains information about each 3D point of the model, identified by its coordinates (x, y, z) and the set of SIFT descriptors defining the point. Using the information extracted from the 3D model and according to the zones dividing our map, we built the image search structures that contain the descriptors of the 3D points belonging to the model.

As we mentioned, to minimize the training time, we recorded a 7'24" (30 frames per second) long video, capturing every detail of our environment. For that purpose, we used of our own designed application that allowed the collection of WiFi and compass measurements during the video recording. Since there are not many objects in the center of the hall, the video was recorded paying attention to the four main walls, with the camera situated at shoulder height. During our tests we noticed that a better reconstruction is obtained when the scene is recorded with the camera oriented towards the walls, including a second round using a 45 degrees angle of view.

Once the video was recorded, and after several tests varying the frame extraction rate, we selected one image every 24 frames, thus obtaining 555 different images (640×480). Around 5% of these images were discarded because of their poor quality. More than 250K feature descriptors were extracted and used to build a 3D model made up by around 34K 3D points. The quality and the quantity of images obtained were good enough to build the 3D model which was used to perform the following tests. Additionally we propose two different variations with the main aim of building more relaxing models in terms of number of descriptors. The first approach tries to reduce the number of descriptors by performing a filtering process during which the descriptors, that were used to define the same 3D point and whose cameras are closer than a specific threshold (100 cm), are removed from the final 3D model. We based this approach on the assumption that the area visualized by these cameras is clearly overlapped and the corresponding descriptors introduce redundancy to the model. As we will demonstrate, using this camera filtering the number of descriptors is reduced to 80K, which leads to a noticeable reduction of the matching time.

Then the *multisensor* integration mentioned in Section 4.3.1, that takes into account the information about the RSSIs behavior and the orientation information obtained from the compass, was used to define several overlapped subzones of the scenario. As it was previously commented, we divided our scenario into four different zones (see Figure 4.10), each one of them made up of 25K of descriptors approximately. During the online phase we can use the information provided by WiFi and Compass sensors to select the zone where the device is supposed to be, and to constrain the image analysis to that tentative zone.

Figure 4.12 shows a comparison of the average time required to perform the matching process using the three different alternatives described to build the image search structures. Using the original set of 250K descriptors the matching process took around 550 ms on average. Applying the described filtering, which reduces the number of used descriptors to 80K, we can save around the 67% of this matching execution time. Finally, taking also advantage of the multisensor approach to define 4 different sub-models, we can reduce the time required to carry out the matching process down to 70 ms on average.

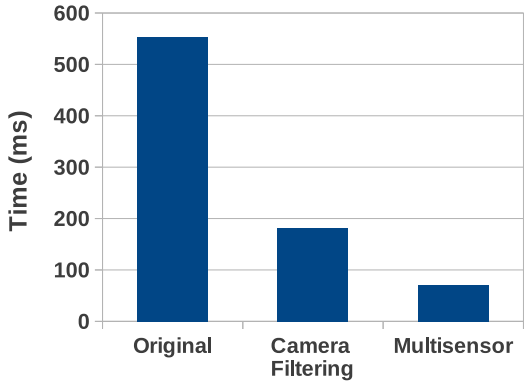


FIGURE 4.12: Performance provided by the different alternatives to build the images search structures.

It is worth mentioning that accuracy was not affected by using these optimizations. The main argument that supports this statement is that filtering descriptors are beneficial when performing a camera resection, since we reduce the possibility of selecting "duplicated" correspondences that imply redundancy to the resection process. Furthermore, the utilization of the zone division approach did not affect the detection of the most useful correspondences since it only constrained the search of the most appropriate submodel according to the previous coarse estimation. Hereinafter, the rest of the experiments described were performed using the most efficient matching version.

Experimental results

In this section we present the experimental results of our proposal. A set of 20 images (640×480 pixels) is our validation data which have been obtained while walking along our scenario using the rear camera of an ASUS EEEPad Transformer TF-101 tablet. Since the camera was previously calibrated, we knew its intrinsic parameters (focal length and aspect ratio). We tested the three camera resection techniques described in Section

4.3.4, executing each combination of parameters 20 times per image. Hence, the shown values are the averaged results of a total of 400 executions.

On the one hand, in the case of DLT and Fiore's algorithms, we evaluated the following parameters:

1. *# Correspondences*: This refers to the number of 2D↔3D pairs used as input for the resection process. Images that can be obtained in the experimental scenario ensured a number of correspondences large enough to test the selected values of this parameter.

2. *Minimum Inliers*: It is the minimum required number of correspondences that must be considered as inliers to stop the RANSAC process and consider a solution as valid.

3. *Maximum RANSAC Iterations*: It is the maximum number of repetitions that the RANSAC process is executed to obtain the transformation matrix that minimizes the reprojection error.

4. *Reprojection Error*: Represents the upper threshold (in pixels) to consider a 2D↔3D correspondence as inlier.

On the other hand, to perform the evaluation of the P3P resection technique, we exhaustively tested a different set of configuration parameters due to its different nature:

1. *Misalignment error*: the threshold ϵ that decides when a 2D↔3D correspondence is considered a tentative inlier. It constrains the maximum alignment error ξ of each individual correspondence with respect to the camera center.

2. *Minimum Inliers*: This refers again to the minimum number of required inliers over the total amount of correspondences used to validate a RANSAC triplet.

3. *Residual value*: the mean residual $\sum_i \xi_i / N$ value obtained after resection (including the non-linear optimization) that indicates the error in alignment for the final number N of inlier correspondences. This is related (though not equivalent) to the reprojection error, and can thus be considered also as a valid metric of the efficiency of the obtained solution.

In both cases, the ranges of values for testing each parameter have been selected after carrying out additional experiments guided by the information obtained from vision-based literature and the study of the efficiency of the correspondences we have been working with. Table 4.5 to Table 4.8 show the different parameter combinations that were evaluated.

# Correspondences		40			50			60		
Minimum Inliers		10	20	30	10	20	30	10	20	30
Iterations	Reprojection Error (px)									
100	4	142,5 - 76,5%	113 - 64%	165,2 - 65%	139,7 - 85,5%	113,9 - 80%	133,2 - 78%	102,8 - 81,5%	85 - 78%	111,6 - 76%
	5	161,9 - 69,5%	152,1 - 68,5%	193,6 - 65%	219,2 - 84%	109,8 - 85%	112,5 - 79%	103,4 - 78%	86,7 - 77,5%	89,2 - 71,5%
	6	193,1 - 66%	172,9 - 74%	172,4 - 61%	114,5 - 79,5%	103,8 - 77%	170,2 - 79,5%	101 - 77,5%	95,2 - 73%	110 - 82,5%
200	4	153,7 - 69,5%	109,7 - 67%	180,6 - 66%	99,5 - 79,5%	118,6 - 81%	149,7 - 81,5%	105,1 - 84%	89,5 - 81%	116,1 - 78,5%
	5	180,7 - 72%	138,6 - 71%	156,1 - 62%	118,7 - 84%	114,3 - 81%	145,3 - 78,5%	109,8 - 80,5%	113,4 - 81%	98,2 - 76,5%
	6	179,2 - 69%	190,8 - 69%	132,5 - 69%	114,4 - 88,5%	112,3 - 83,5%	125,7 - 79,5%	86 - 77,5%	105,7 - 80,5%	116,1 - 73,5%
300	4	225,6 - 70,5%	137,1 - 69%	152,4 - 65,5%	116,8 - 80,5%	114,8 - 81%	176 - 83%	108,4 - 77,5%	95,1 - 80,5%	94,8 - 76,5%
	5	272,1 - 74%	178,2 - 67%	133 - 68%	115,9 - 85%	97,6 - 81,5%	120,8 - 79%	87,9 - 80,5%	65,3 - 78,5%	129,5 - 78%
	6	154,1 - 72,5%	156,8 - 66,5%	245,9 - 69%	111,5 - 87%	126,4 - 85,5%	154,9 - 76%	105,6 - 80,5%	99,4 - 76,5%	115,6 - 76%

TABLE 4.5: Results of the resection process using DLT (with the Early-Detention version of RANSAC) and different configuration parameters. Estimation Error (cm) - Successful camera resection cases (%). Red background cell represents the most accurate option and the configuration that gets the best trade-off between accuracy, performance and reliability.

# Correspondences		40			50			60		
Minimum Inliers		10	20	30	10	20	30	10	20	30
Maximum RANSAC Iterations	Reprojection Error (px)									
100	4	85,7 - 100%	91,4 - 96%	81 - 90%	81,9 - 98%	74,5 - 97,5%	60,5 - 88%	86,9 - 94,5%	60,1 - 93%	61,8 - 86%
	5	78,3 - 100%	73,4 - 97%	70,6 - 88%	86,7 - 96%	76,4 - 96%	61,4 - 85,5%	78,2 - 93,5%	70,2 - 93%	55,7 - 86%
	6	87,7 - 99%	90,5 - 97,5%	76 - 90%	86,9 - 96%	77,9 - 94%	60 - 86%	81,3 - 94%	75,7 - 93,5%	66,2 - 87%
200	4	82,9 - 100%	70,2 - 97,5%	68,2 - 89,5%	78,3 - 98%	78,5 - 97,5%	64,6 - 87%	74,6 - 97%	59,3 - 94%	69,8 - 86,5%
	5	66,4 - 99%	63,7 - 99%	56,8 - 89%	89,3 - 95,5%	86 - 97%	72 - 86,5%	83,3 - 93,5%	68,2 - 93,5%	57,7 - 85%
	6	86,9 - 98%	91,3 - 99%	90,5 - 88,5%	77 - 95%	79,4 - 95%	72,2 - 86%	75,1 - 94,5%	76,2 - 94%	61,5 - 87,5%
300	4	82,8 - 99,5%	89,9 - 99%	61,3 - 90%	69,5 - 97,5%	67 - 95%	56,7 - 87,5%	79 - 94%	61,5 - 93,5%	57,7 - 86,5%
	5	73,7 - 98,5%	73 - 99%	53,5 - 90%	78,3 - 96,5%	79,9 - 96%	57,5 - 85%	74,5 - 94,5%	68,6 - 93,5%	66,7 - 86,5%
	6	93,2 - 98,5%	76 - 99%	72,1 - 90%	73,3 - 95%	77,6 - 94,5%	65 - 85%	84,7 - 93,5%	71,5 - 93%	59,6 - 90%

TABLE 4.6: Results of the resection process using DLT (fully trying maximum number of combinations in RANSAC) and different configuration parameters. Estimation Error (cm) - Successful camera resection cases (%). White shaded cell represents the most accurate option. Red background cell depicts the configuration that gets the best trade-off between accuracy, performance and reliability.

# Correspondences		40			50			60		
Minimum Inliers		10	20	30	10	20	30	10	20	30
Maximum RANSAC Iterations	Reprojection Error (px)									
100	4	19.5-100%	10.11-96%	9.0-90%	25.2-100%	11.60-97%	9.6-90%	39.9-99%	15.6-96%	9.4-90%
	5	14.4-100%	15.1-98%	10.8-90%	25.8-100%	19.0-98%	9.8-90%	32.5-99%	17.0-95%	10.6-91%
	6	16.3-100%	13.9-99%	10.5-90%	27.2-100%	19.3-98%	10.1-90%	37.0-100%	15.0-95%	15.0-91%
200	4	10.8-100%	9.5-97%	8.6-90%	20.0-100%	17.9-99%	10.1-90%	29.1-99%	13.6-98%	9.2-90%
	5	14.6-100%	12.3-99%	10.7-90%	15.9-100%	17.3-99%	9.7-90%	18.1-99%	22.0-97%	10.9-92%
	6	14.0-100%	12.9-100%	10.0-90%	15.9-100%	12.0-100%	10.2-90%	32.5-100%	17.6-98%	12.4-92%
300	4	12.4-100%	10.4-97%	9.1-90%	15.3-100%	11.5-99%	10.2-90%	19.3-100%	14.4-98%	10.0-90%
	5	14.1-100%	11.9-99%	10.6-90%	11.6-100%	15.3-100%	9.5-90%	23.4-100%	13.2-98%	11.0-92%
	6	12.3-100%	11.6-100%	9.9-90%	14.7-100%	16.1-100%	10.0-90%	19.8-100%	13.0-99%	12.5-93%

TABLE 4.7: Results of the resection process using Fiore's algorithm and different configuration parameters. Estimation Error (cm) - Successful camera resection cases (%). White shading cell represent the most accurate option. Red background cell depicts the configuration that gets the best trade-off between accuracy, performance and reliability.

Misalignment Error	0,01			0,03			0,05		
Minimum Inliers	10	20	30	10	20	30	10	20	30
Residual Value									
0,0001	25,5-71%	13-67%	12,6-68%	25,9-41%	14,7-45%	11,5-41%	22,6-30%	17,8-31%	13,1-29%
0,0005	14,8-100%	16-98%	14,6-90%	17,7-98%	15,7-98%	14,7-90%	22-96%	18,3-96%	17,7-89%
0,001	16,8-100%	17,1-97%	13,3-90%	18,6-99%	15,8-100%	15,7-91%	21,9-98%	20-100%	17,7-91%
0,0015	13,9-99%	12,7-98%	12,8-90%	19,1-100%	17,1-100%	14,7-90%	20,6-99%	18,9-99%	19,9-91%
0,002	12,1-100%	12,2-98%	10,8-89%	11,1-100%	12-98%	10,4-90%	13,9-95%	13,6-98%	14,1-91%

TABLE 4.8: Results of the resection process using P3P and different configuration parameters. Estimation Error (cm) - Successful camera resection cases (%). White shading cell represent the most accurate option. Red background cell depicts the configuration that gets the best trade-off between accuracy, performance and reliability.

Performance: First of all we will discuss the factors that influence the resection time. Considering DLT and Fiore's, the most demanding step in terms of time is the RANSAC process execution, which is directly influenced by the number of iterations which are carried out to find the best set of inliers. As we show in Figure 4.13, the higher the number of iterations, the longer it takes to execute the RANSAC process. Considering the negligible time required to execute the QR-Decomposition (DLT) or the Fiore's algorithms, which is 1 and 3 milliseconds respectively, we conclude that most of the time needed to perform the camera resection is used at RANSAC. Therefore it gets crucial to minimize the number of iterations to boost the full process.

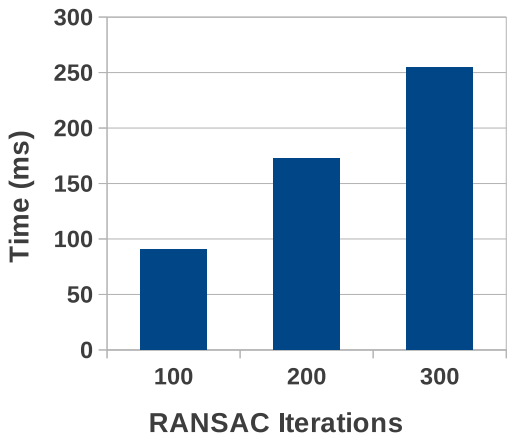


FIGURE 4.13: Comparison of times required to carry out the RANSAC process according to the number of iterations performed.

In an attempt to reduce the time required for RANSAC, we evaluated the use of the so-called *early-detection* (ED) version of the RANSAC process for the case of the DLT algorithm. It was based on the idea that once the threshold of *minimum inliers* has been exceeded, it might not be necessary to continue the RANSAC process, since the already obtained camera matrix P might be sufficient to estimate the camera pose accurately. Introducing this modification, we clearly reduced the processing time for the full camera resection process, as shown in Figure 4.14. In this case, the parameter *Maximum RANSAC Iterations* did not have any influence over the final result since most of the times the threshold of the minimum number of inliers was exceeded before completing all iterations.

However, as it can be observed from the comparison of ED test results (see Table 4.5) with Non-ED results (see Table 4.6), in general terms the utilization of this ED version

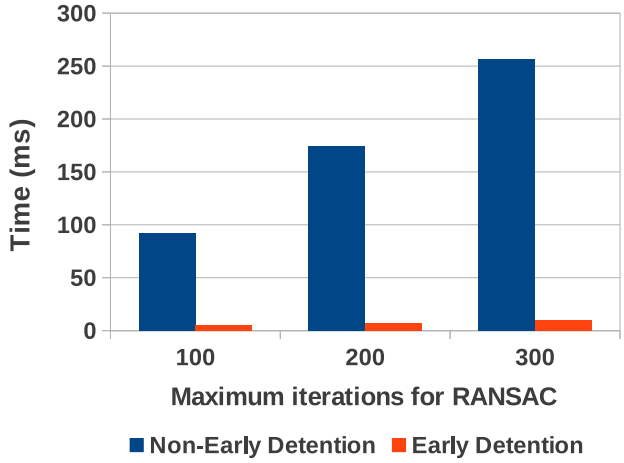


FIGURE 4.14: Time required to carry out the full resection process, comparing the utilization of the ED and Non-ED versions of RANSAC using DLT.

of the RANSAC had an major negative influence on the final accuracy obtained. Despite its good performance (saving up to 60% of resection processing time), we finally decided to discard this option because it clearly deteriorates the obtained accuracy.

We also evaluated the performance of the P3P approach in a posteriori phase. Since it was deployed as an alternative to Fiore's for the camera calibrated case, we compared these two techniques. As Figure 4.15 shows, the use of P3P implies an vast improvement on the time required to perform the full camera resection, taking now less than 10 ms. The main reason is the reduced time required to run the RANSAC process because of the lower number of correspondences needed (only 3). In this case, we use the parameter *Maximum RANSAC Iterations* = 100.

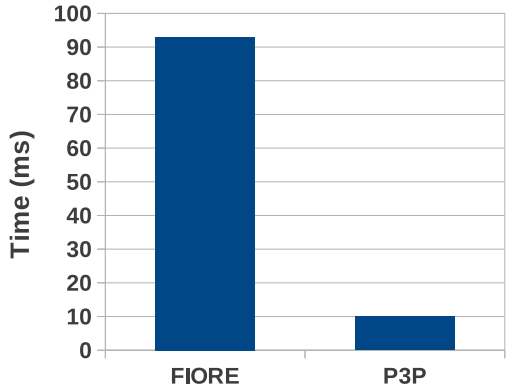


FIGURE 4.15: Time required to carry out the full resection process using Fiore's and P3P algorithms.

Figure 4.16 shows the time required to complete an entire cycle using P3P. It details the transmission time of sensor data from the smartphone, the required time at the server to perform the SIFT extraction, to run the matching process, and to apply the resection technique. Since the time required to perform the RSSI analysis is around 1.5 milliseconds, it has been omitted in Figure 4.16 because of its unnoticeable influence. As it can be observed, the time needed to complete the whole cycle from sensor data transmission to 3D estimation is around 250 milliseconds on average.

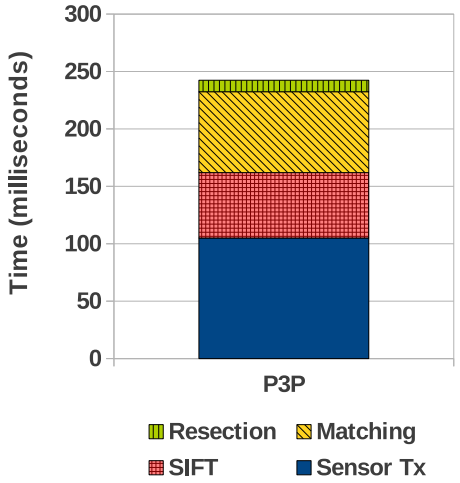


FIGURE 4.16: Time required by the different tasks involved into the entire cycle, from the sensor data transmission to the position estimation.

Accuracy: The evaluation of the accuracy provided by this type of LBS is not trivial to measure because of the existing problems to define a reliable ground-truth. In our case, accuracy results were obtained using as ground-truth the positions of the cameras assigned by VisualSfM using the built 3D model. We are aware that it is not a precise estimation, but considering that the 3D model was carefully adjusted to the real world coordinates, it can be considered as a good metric of the efficiency of the obtained solution during the on-line phase. Camera rotations where not quantitatively assessed, however, both accuracy of the camera center and rotation obtained were quantitatively evaluated during a posterior phase making use of an Augmented Reality (AR) application.

As we can extract from the results shown in Table 4.6, DLT can be considered an acceptable option only in those cases in which it is necessary to perform a camera autocalibration. It allows us to obtain an accuracy around 66 cm of estimation error, which could be sufficient depending on the final application requirements.

Regarding Fiore's, there are several sets of parameters providing, on average, an estimation error under 15 cm. In this case, the choice of the most appropriate configuration depends on the associated processing time which is influenced by the number of iterations executed, as it was previously described. Therefore, using the set of parameters that provides the best trade-off between accuracy, performance and reliability (red background cell in Table 4.7), we are able to provide an accuracy down to 15 cm of estimation error for the 100% of tested cases, spending around 145 milliseconds to run the full resection process.

Despite the fact that this option implies a slight loss of accuracy in relation to the most precise one (white background cell in Table 4.7), it means a reduction of the execution time of approximately 37% regarding to those cases where the value of *Maximum RANSAC Iterations* parameter was equal to 200, and around 53% better than performing 300 iterations. Moreover it implies a higher reliability, since in 100% of executions the technique was able to provide an accurate position estimation. It is worth noting that while this selected set of parameters works properly for our environment, it could differ from one scenario to another.

However, as it can be observed in Table 4.8, where we show the results obtained using the P3P algorithm, we were able to provide an error estimation down to 12 cm on average and to maintain the same reliability (100% of successful cases), making use of the parameter set that corresponds to the red background cell. Furthermore it is important to remind the reader the notable improvement in terms of execution time that the utilization of this technique implies for camera precalibrated situations. Once again, it is important to mention that the choice of the most appropriate parameter configuration would depend on the specific characteristics of each environment.

Figure 4.17 shows a summary of the data presented in Tables 4.5 to 4.8, comparing the accuracy obtained by each one the resection techniques that were evaluated.

Augmented Reality: Finally, in order to test this approach from a qualitative perspective, we developed a prototype app for Android devices based on OpenGL ES 2.0 which is able to display augmented reality based on the precise estimations of the full 6 degrees of freedom: (X, Y, Z) position and (α, β, γ) Euler angles for the rotation matrix. We display a wire-frame model on top of the real world to visually check the accuracy

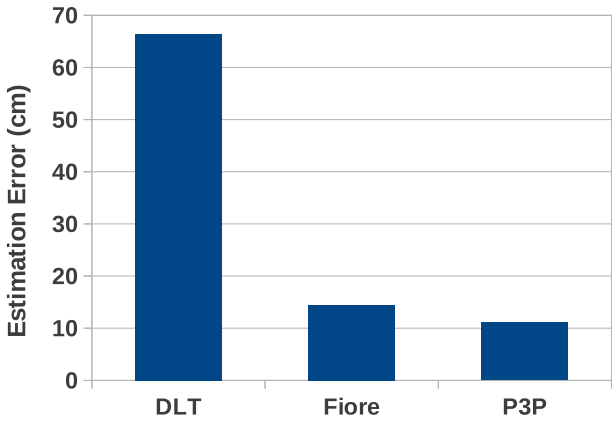


FIGURE 4.17: Comparison of the accuracy obtained making use of the different resection techniques evaluated.

of our estimations. We added virtual banners to the doors in order to display information of interest. Figure 4.18 shows a snapshot of the prototype which demonstrates the accuracy obtained.

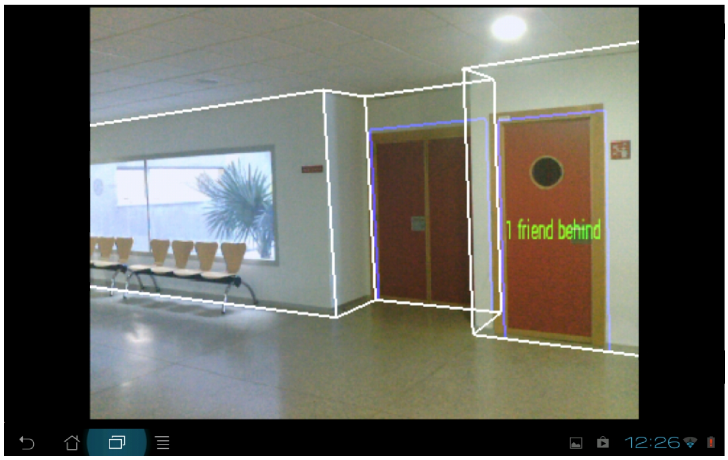


FIGURE 4.18: Snapshot of the Android AR app. It shows a wire-frame model (including virtual banners) on top of the real world, which demonstrates the accuracy of our system.

4.4 Summary

As emphasized during the chapter, the integration of vision-based approaches was crucial to accomplish a robust and accurate location service. In our initial proposal, we introduced a simple place recognition solution based on image matching. This first approach to the images analysis allowed us to demonstrate their suitability to improve the system accuracy. Real-time experiments provided good results for still and motion scenarios in

terms of accuracy and performance, achieving a cell hit rate (including adjacent cells) around 94% in motion situations. Though dependent on the layout of the environment, we are aware that this proposal is not precise enough to support localization-based services requiring a high accuracy. However, it seems a practical solution for those environments in which, due to their physical characteristics, there is no possibility to deploy more sophisticated systems and the use of images would be sporadic.

Regarding our second proposal, presented in Section 4.3, we notably improved both the scalability and the precision of the training phase, not only eliminating the need to discretize the environment at a given granularity, but also improving the quality of the location estimations. As demonstrated, several advantages are obtained using a reduced number of training images from a video recording, a key fact that makes the training phase definitely much easier to be completed. The introduction of SfM techniques means an important improvement on the system accuracy, since it facilitates the reconstruction of precise 3D models of the environment, allowing the utilization of camera resection techniques to estimate the full 6 dof that indicate the camera pose and its rotation. Precise locations were obtained with an average error down to 12 cm with the additional bonus of estimating the rotation with high precision.

Moreover, both performing a simple image matching or by incorporating SfM techniques, we demonstrated that using the multisensor integration (WiFi, Compass and Accelerometer) we remarkably augmented the system scalability and performance. In both cases the time required to complete an entire cycle was notably reduced introducing this approach, taking around 525 ms and 250 ms respectively, making this second proposal able to support those applications that require high accuracy and rapid response, such as augmented reality applications. According to the obtained results, we find our refined approach a valuable contribution for this kind of location-based services, since we have obtained a good trade-off between a fine-grained accuracy and an acceptable response time using data from multiple sensors.

It is also worth mentioning the main drawbacks of our approach. As we introduced, SIFT features used alone can fail to disambiguate while building the 3D models, requiring the manual intervention of an operator to ensure the reconstruction of feasible models. This can clearly complicate the deployment of our solution in some environments. Nonetheless, the multisensor approach has demonstrated that it provides mechanisms to work

in difficult scenarios such as those presenting similar areas from a visual perspective (floors, rooms). This is an important issue that will be addressed in the next chapter.

Finally, it is important to mention our concern regarding to the privacy issues. In this kind of systems privacy is one of the main issues, specially when working with images. Nowadays, transmission of images is a requirement due to the limitations of current tested smartphones to perform the extraction and the matching of useful image features. However, some manufacturers are launching new smartphones with built-in GPUs, like NVIDIA TEGRA 5, that will be able to perform that processing locally. This is especially interesting not only to reduce the workload on the localization server, but also from the point of view of the privacy, since images would not have to be transmitted. Some on-going works are currently focused on solving this issue.

Chapter 5

Improving system training using multisensor information

5.1 Introduction

Continuing with the integration of multiple sensors, in this chapter we present several key improvements focused on providing methods that facilitate the training phase, building more accurate 3D reconstructions and thus obtaining a better accuracy during the localization stage. As mentioned in the previous chapter we identified some issues regarding the reliability of the process defined to build the 3D maps. These issues were related to the appearance of repetitive visual structures and objects throughout the modeled scenario, which produced erroneous models and required the manual intervention of an operator to minimize these flaws. Moreover, in these conditions our proposal suffered from serious scalability problems when modeling environments larger than our experimental scenario.

With the main aim of solving this important handicap, we propose a solution to perform a guided training phase with the following aims:

- To estimate the position where the operator collects the sensors measurements during the training.
- To define a method to constrain the image matching list used by SfM depending on the camera estimated positions.

- To obtain an accurate 3D representation of the environment as a result without requiring manual intervention during the SfM process.

To accomplish this work, we firstly introduce the use of *inertial measurement units* (IMU) to generate accurate 3D models in a mostly automatic and unsupervised way. We define a specific procedure where an operator carrying a tablet or smartphone walks around the indoor environment taking time indexed pictures that will be used to build the 3D model, among other sensor information. Once the walk is finished, we process the inertial sensor measurements to infer rough locations of the places from where each image was collected. This method only requires an approximate vectorial map of the scenario as an input. Then we carry out a post-processing stage where images are automatically geo-tagged using the inferred pathway. This approximate spatial information will be paramount to rule out comparisons between images taken from very different angles or distant positions, thus minimizing the reconstruction problems due to the existence of visually similar structures.

Afterwards, in order to validate the usefulness of the automatically generated 3D model, we also analyze the accuracy of our location estimations during the on-line phase using the camera resection procedures that were described in the previous chapter. Though the improvement of the system accuracy is not a key goal now, to perform this assessment we do several experiments using a representative dataset of images previously geo-tagged. Finally, making use of an AR application prototype we validate the quality of the location estimations (camera center and rotation) and demonstrate the suitability of our system to support applications that require high accuracy and rapid response time.

The rest of the chapter is organized as follows. Section 5.2 summarizes the proposal and gives an overview that facilitates the comprehension of the rest of the chapter. Section 5.3 depicts some details of the scenarios where experimental tests were conducted. Afterwards, in Section 5.4 we mention the particulars of our utilization of inertial and orientation sensors to infer operator movements. Then Section 5.5 defines the method on how to carry out a camera filtering that supports the generation of 3D models in an unsupervised way. Finally Section 5.6 presents some experimental results and Section 5.7 summarizes the major achievements accomplished.

5.2 Proposal overview

Here we summarize the suggested refinement. This new solution also takes advantage of the multisensor approach to simplify the training process, making it more robust from the point of view of the reliability provided to perform the 3D reconstructions. As in the previous proposal, the multisensor integration speeds-up the localization phase by limiting the amount of information to be examined during image analysis.

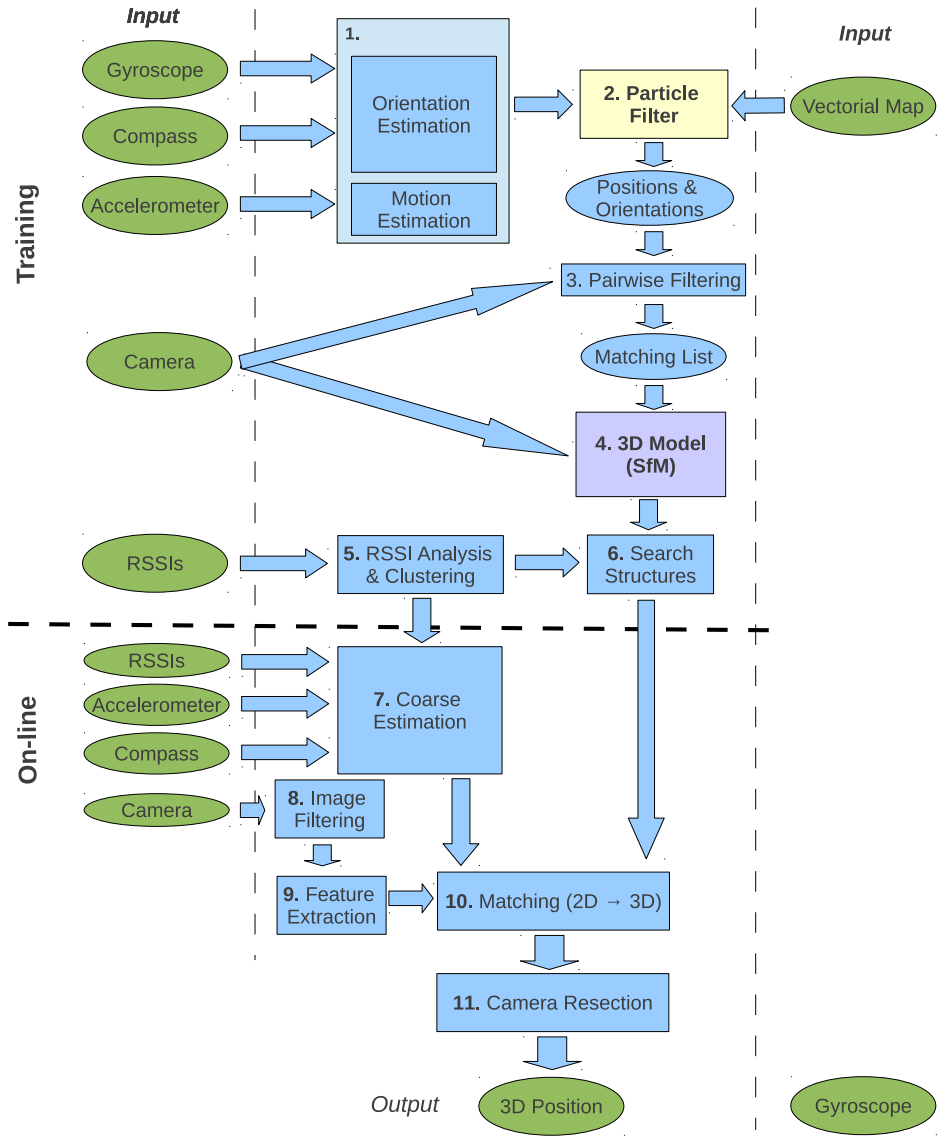


FIGURE 5.1: Overview of current proposal. It includes all steps to be taken during training and on-line phases.

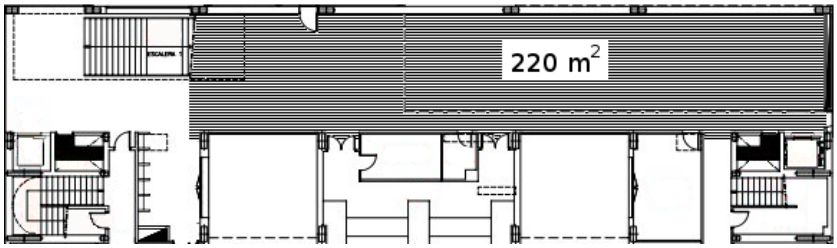
An overview of the current system is shown in Figure 5.1. During an initial training phase, images, WiFi RSSIs and measurements from IMUs are captured along the entire scenario. On the one hand, we use the data from the inertial sensors (Step 1 in Figure

5.1) in order to perform a coarse-grained estimation of the locations where the images were taken from. That estimation is based on a particle filtering procedure (Step 2) that makes use of both the data from the inertial sensors and a rough vectorial map of the environment. The result is used to calculate a list of pairwise camera candidates (Step 3) used to guide the image matching process during the reconstruction, avoiding comparisons between images which were taken using different angles or from distant positions. This constrained matching process will allow the automatic generation of very precise 3D models (Step 4), mostly eliminating the need of human supervision during the reconstruction process. Finally, using the fingerprinting and clustering techniques described in previous chapters (Step 5 in Figure 5.1) we divide again the scenario into different zones according to the characterization of the 802.11 signals. As a consequence, we obtain clusters of physical points where signals show a similar behavior, which will be used to partition the generated 3D model into different submodels (Step 6).

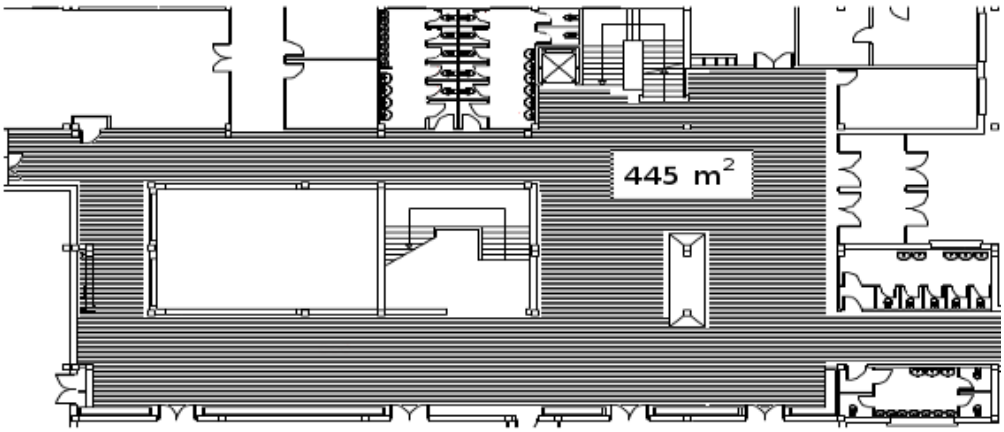
The online phase has not changed with respect to our previous proposal. Firstly, we determine a coarse-grained geographical zone where the device is located using the observed RSSI (Step 7). Then during the final steps (Steps 8 - 11), the images are analyzed to determine their usefulness, to extract the set of feature descriptors and to obtain the 2D-3D correspondences used as input for the camera resection procedure. The 3D position obtained is refreshed according to the measurements of the gyroscope in order to provide a comfortable refreshment rate and to improve the user experience.

5.3 Experimental environment

This section describes the characteristics of the real environments where the experiments described in this chapter were conducted. We evaluated our proposal in two different scenarios. The first one corresponds to the ground floor of the Faculty of Computer Science (Scenario 1), see Figure 5.2(a), where the experiments described in previous chapter were also carried out. The second testbed is the main floor of Faculty of Veterinary Science (Scenario 2), see Figure 5.2(b). These scenarios cover an area of 220 m^2 and 445 m^2 respectively. In order to compare current results to those obtained in previous chapter, the experiments described throughout this chapter were conducted in Scenario 1. The Scenario 2 was modeled in a posterior phase with the main aim of validating our proposal.



(a)



(b)

FIGURE 5.2: (a) Scenario 1: Ground floor of the Faculty of Computer Science ; (b) Scenario 2: Main floor of the Faculty of Veterinary Science.

To deploy this system we maintain the client-assisted architecture described in the previous chapter. The server used for these experiments corresponds to the same described in Section 4.3.1. During the training phase we used a custom sensor capture application running on a Samsung Galaxy Tab 7.0 (see snapshot in Figure 5.3). This application is an evolution of the training app mentioned in the previous chapter. It was developed to acquire all the relevant sensors data (WiFi, camera, accelerometer, gravity, gyro and compass), which are transmitted to the core server for its analysis off-line. The camera of this device was calibrated and its focal ($f = 615$ pixels) for images of size (640×480) was estimated. The operator performed several walks throughout the scenario, following a previously defined path according to a predefined motion behavior. During these walks accelerometer, gyro and compass were measured at a frequency of 50Hz approximately. Additionally, virtual sensors like gravity and linear acceleration sensors were also monitored. Analyzing these readings we were able to detect whether the device was in motion or remained still, which also helped us in determining the best time instants to capture individual images. RSSIs are obtained performing passive scans of the wireless network.

It is important to mention the necessity of using a training device supplied with quality inertial sensors providing good measurements to obtain the desired results.

During the posterior on-line phase we made use of an Augmented Reality (AR) application (see Figure 5.15) able to collect data from different sensors (WiFi, compass, accelerometer and camera), to request the current position, and to display the augmented reality according to the estimated position.

FIGURE 5.3: Main screen of our training application. The left side of the screen shows information from sensors measurements in real time as the operator moves throughout the scenario. On the right side it is shown the image visualized by the camera as well as an updated evolution of the particle filtering within the map.

5.4 Motion estimation and particle filtering

In this section we describe the fundamentals of our basic motion model that supports the training phase. We first present the method that has been developed in order to detect motion situations as well as the inference of movement orientation. Afterwards, we describe the utilization of the particle filtering approach in order to fit the estimated movements to the real layout of the environment where we are moving around.

5.4.1 Motion and orientation estimation

It is a well known fact that the accelerometer usually integrated in mobile devices is relatively noisy. The great magnitude value of the always present gravity ($9.8m/s^2$) with

respect to interesting accelerations induced by user movements is indeed a problematic issue. In fact, the problem of trying to use a direct double integration of the acceleration readings in time, in order to obtain the velocity vector (first integration) and then the movement vector (second integration), is that this double integration enormously amplifies this noise as a cubic function of time. This renders the result virtually useless in less than typically two or three seconds, at best, depending on a number of factors.

The problem of drifting issues when integrating the accelerometer measures for long periods of time has been addressed satisfactorily by navigating in open loop for only less than a second at a time, taking advantage of the fact that, when a person walks, the feet periodically experiment a stationary stance phase of approximately 0.5 seconds in every step [62]. An intelligent integration scheme, that applies *zero-velocity updates* (ZUPT) when detecting this stance phase, together with a carefully designed *Extended Kalman Filter* [172], is able to accurately estimate the position of a pedestrian for relatively long walks. Nevertheless, it works with higher quality inertial sensors than those commonly found in typical smartphones and tablets, and requires the sensor to be mounted in the shoe of the operator, in order to take advantage of the ZUPT trick. These drawbacks clearly limit the applicability of this technique in our case, where the operator must hold the tablet in their hands to take pictures of the environment.

Other existing approaches involve detecting periodic patterns in the acceleration magnitude signal, corresponding to user steps, thus using the device as a pedometer. In these cases, incorporating information from the compass, the approximate direction of the user can be determined and, using a particle filter and a known map of the environment, we can adequately correct the generated trajectories [147]. Again, a basic requirement in here is that users have to put their phones in the pocket or somewhere else where the periodic movement pattern is clearly noticeable. Once more, this is not applicable in our case, as the operator has to take pictures with the device.

In this work we somehow combine these two approaches in order to adapt them to our needs. What we propose is to limit the kind of movements that the operator has to perform while taking pictures of the environment, in a way that it:

- Makes it easy to predict movements.
- Ensures that pictures are taken in the best imaging conditions.

- Simplifies the operator's job, who has to take hundreds of pictures from different positions on the environment during his walk.

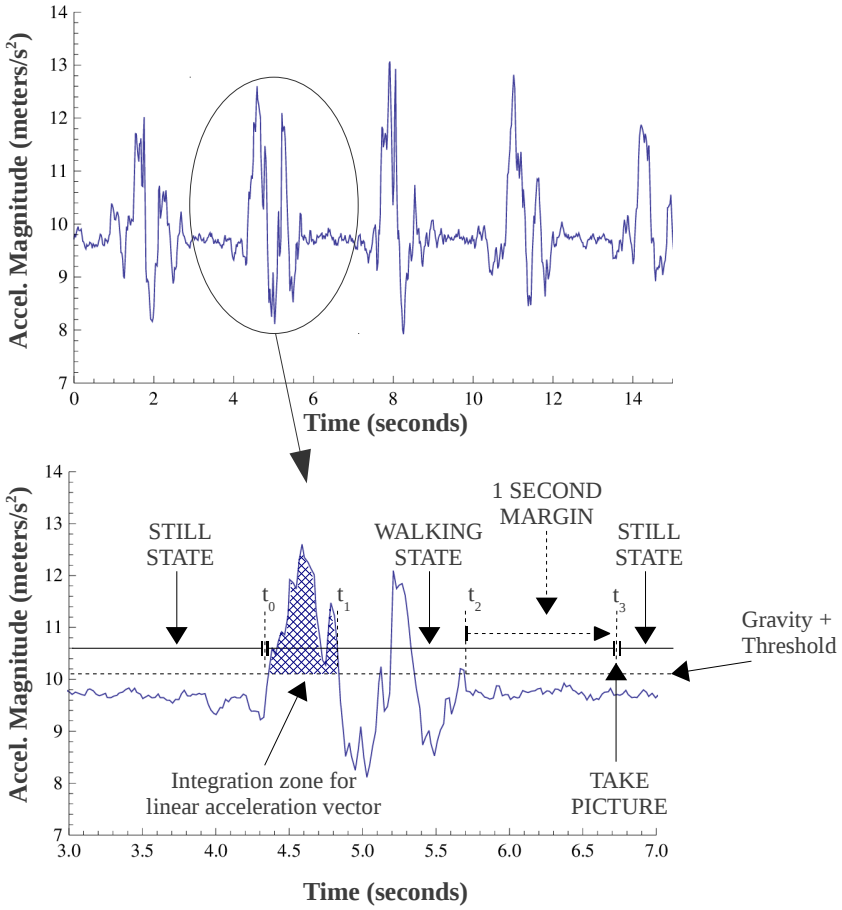


FIGURE 5.4: Acceleration magnitude analysis.

In this regard, we implemented an algorithm which expects a movement pattern in which the operator stands still, takes a picture of the environment, moves to a nearby position (typically one meter away or so), and then stands still again to take a new picture, always holding the tablet or smartphone. Under these conditions, we will describe a simple procedure that will allow us to roughly estimate the approximate movement performed by the operator between every two consecutive pictures. This relative movements, though certainly noisy, and suffering a considerable drift when integrated in the long term, will be the basis for a posterior particle filter, that will correct these measurements adapting them to the scenario map, much in the style of the approach presented in [147].

Our basic movement model is summarized in Figure 5.4. As it can be observed, we analyze the magnitude $|\overrightarrow{a}|$ of the acceleration vector $\overrightarrow{a} = (a_x, a_y, a_z)$ in time, sampled

at $50Hz$ by the device, and then we consider two main states of the operator, *Still* and *Walking*. The logic for changing from one to the other is quite simple: as long as the accelerometer signal exceeds the gravity $|\overrightarrow{g}| = 9.8\ m/s^2$ by a predefined small threshold (see instant t_0 in Figure 5.4), we immediately consider that the operator is in the walking state.

Nevertheless, during the walking state there may be short periods of time where $|\overrightarrow{a}|$ falls below that threshold, without its necessarily meaning that the operator has stopped. Thus, the application only gets back to the still state as long as $|\overrightarrow{a}|$ keeps under the threshold value for a period of at least one second (from t_2 to t_3 in Figure 5.4). At that precise instant, the application automatically takes a picture. This automatic process helps in taking clear and sharp images, as the operator does not have to interact with the application interface, thus ensuring that the tablet/smartphone is fairly still when the picture is taken.

Now we have to estimate the relative movement performed by the operator during the walking period. As already said, double integration of the linear acceleration vector $\overrightarrow{a} - \overrightarrow{g}$ is only reliable during very short periods of time. For that reason, we opted for integrating it only until the magnitude $\overrightarrow{a}$ falls under the threshold for the first time (from t_0 to t_1 in Figure 5.4) to avoid serious drifting issues. Of course, refusing to integrate the whole interval corresponding to a full walking state period makes the magnitude of the estimated displacement vector absolutely useless. On the contrary, the direction of movement is acceptably estimated. Thus, as output in this stage we only obtain a (unit norm) direction vector $(\Delta x, \Delta y, \Delta z)$, leaving the stride length estimation to the subsequent particle filter:

$$\overrightarrow{\Delta x_{dev}} = (\Delta x, \Delta y, \Delta z) = \frac{\iint_{t_0}^{t_1} (\overrightarrow{a} - \overrightarrow{g}) dt}{|\iint_{t_0}^{t_1} (\overrightarrow{a} - \overrightarrow{g}) dt|} \tag{5.1}$$

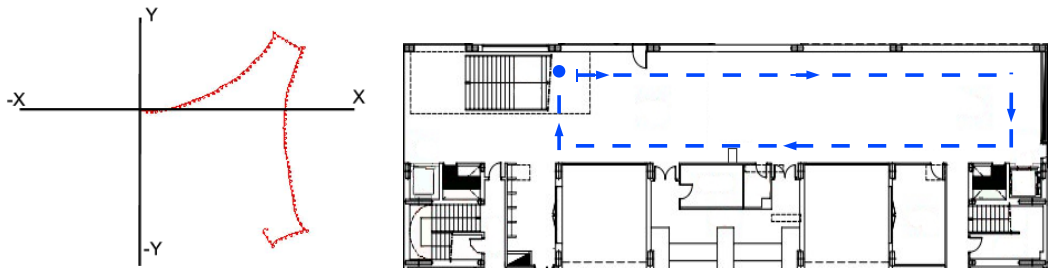
where t_0 and t_1 determine the integration interval of time, as stated above, and shown in Figure 5.4.

This vector has to be corrected, since it still refers to device coordinates, using an absolute rotation with respect to the real world coordinates. For this purpose, many mobile devices do in fact integrate compass, gravity and gyroscope readings to get a virtual sensor that provides a filtered output rotation $R_{dev \to wrl}$. This sensor integration

prevents typical drifting of the gyroscope estimation alone in the long term, by taking advantage of the always available external references of the gravity vector and magnetic north. In our system we use this type of virtual sensor, which in our particular training device behaves quite well in practice. Thus, our final estimation of the relative direction vector in world coordinates will be:

$$\overrightarrow{\Delta x_{wrl}} = R_{dev \to wrl} \cdot \overrightarrow{\Delta x_{dev}} \tag{5.2}$$

Considering the Scenario 1, Figure 5.5(a) represents the raw direction estimations of the movements performed by the operator while covering the path shown in Figure 5.5(b) . As mentioned, the stride length estimation as well as the adjustment to real map of the scenario are delegated to the particle filter.



(a) (b)

FIGURE 5.5: (a) Raw estimation corresponding to a closing loop walk, as obtained by the motion model. (b) Real path that corresponds with the raw estimation.

5.4.2 Particle filter

Once we have defined our methodology for detecting steps or, more precisely, *unitary* movements between picture taking positions, we have to define how to combine that information with the knowledge of the map in order to correctly track the operator movements. The alternative we selected was the implementation of a particle filter based on the proposal made by Thrun et al. in [172]. In order to filter out long term drift errors in the raw input trajectory, it uses spatial restrictions of the a priori known indoor cartography, as well as other restrictions of the path that the operator must cover.

A particle filter is a nonparametric implementation of the Bayes filter and is frequently used to estimate the state of a dynamic system. This technique was initially proposed

by N. Gordon et al. [69] in 1993. The idea behind the particle filter is to represent a posterior distribution via a set of state samples (or particles), where each particle represents one potential state the system might be in. The implementation of the particle filter approach is specially recommended for those scenarios where there is no a priori knowledge about the initial state, but also when multi-modality situations can appear due to the specific characteristics of the problem to be solved. The ability to model multi-modal distributions by the set of samples is an important advantage of this technique. The algorithm can be roughly divided in four different stages:

1. The first step corresponds with the initialization phase, where a finite set of M particles is generated. In case of lack of previous knowledge about the starting situation, each one of these particles is assigned a random state. This initialization must ensure a homogeneous particles distribution throughout the map.

2. During a second step, the different particles belonging to the current set are processed using the current raw input measures in order to evolve them to a hypothetical new state.

3. Then, at the third step the so-called importance factor (or weight) of each particle is computed. This weight determines the usefulness of each particle and it is used during the following resampling process that produces the next set of particles.

4. Finally, a resampling step takes place. The idea of the resampling is to remove the trajectories that have small weights or do not fit with the restrictions of the map. The main aim is to focus the next generation on trajectories that are still plausible. After the resampling, the new set of particles is used as input for step 2. Steps 2 to 4 are iteratively repeated until no more inputs are available.

Figure 5.6 illustrates the overall functioning of our particle filtering algorithm, as working in Scenario 1. Observe that our vectorial map modeling our $220m^2$ faculty hall includes a virtual rectangle in the center which, though inexistent in practice, models a zone avoided by the operator on their walks, and helps to get a much faster convergence of the particle filter to a mostly monomodal state while keeping the sample size low.

The particle filtering process can be summarized as follows:

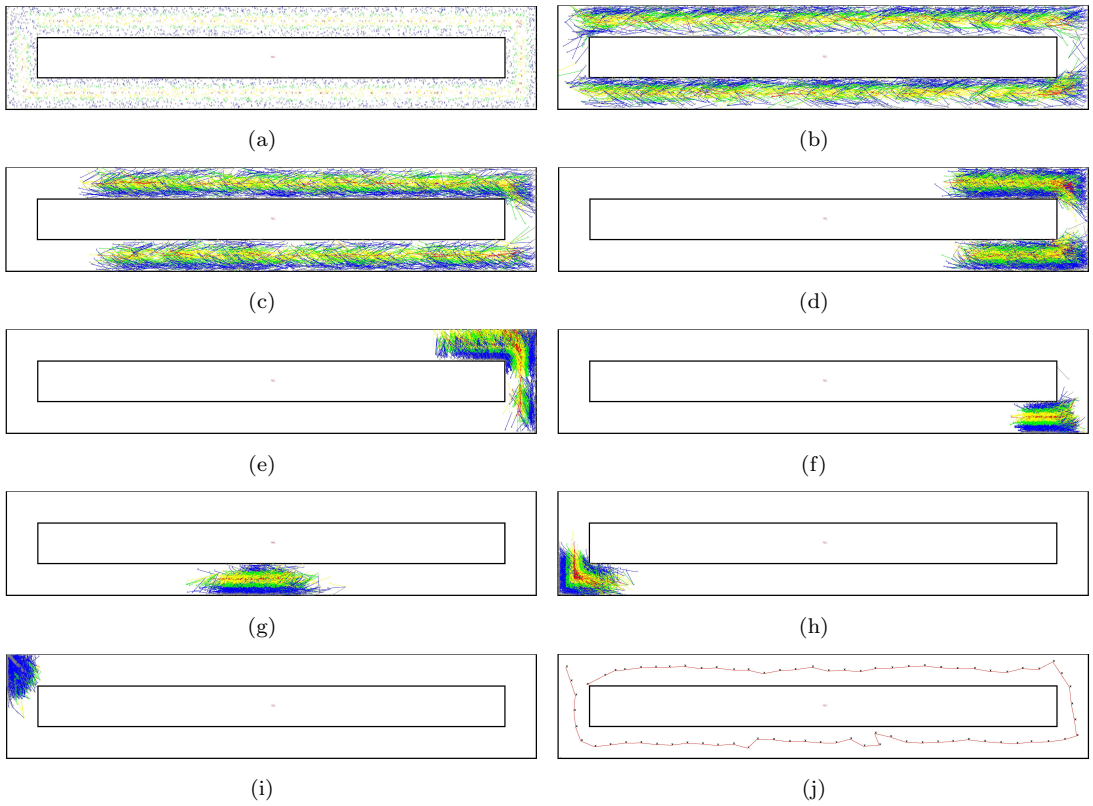


FIGURE 5.6: Particle filtering evolution process corresponding to a walk that cover the path indicated in Figure 5.5(b) throughout Scenario 1. (a) Initial particle distribution; (b - h) Intermediate situations; (i) Final particle set; (j) Corrected path estimation after performing a backward belief propagation.

During an initial phase, particles are generated using the map to create a uniform distribution, representing a complete uncertainty on the initial position (Figure 5.6(a)). The set of particles of a posterior distribution in a given time instant t is denoted as $P_t = \{p_t^1, p_t^2, ..., p_t^M\}$, where each p_t^m (with $1 \leq m \leq M$) is a particular instantiation of the state, that is, a hypothesis as to what the true world state may be at time t. M represents the number of particles. It is often a large number but it may vary depending on the characteristics of the environment and the specific problem we are trying to solve. Moreover, it must ensure a complete and uniform distribution of particles throughout the entire scenario. In general the more samples that are used, the better the approximation, but the involved computational load also increases accordingly.

Once this set of initial particles P_0 is defined, it must evolve according to the sensor readings (Figures 5.6(b) to 5.6(h)). In our case, this corresponds to the unitary displacement vector estimated as described in the previous section. Each particle has associated the following information: $p^i = (x_i, y_i, s_i, o_i)$, with (x_i, y_i) being the associated 2D position,

s_i the stride length and o_i the orientation angle.

In order to accommodate slight variations in the movement of the operator, we allow s_i values in an interval of $[100, 150]$ *cm* for the particles, and add a random error uniformly distributed in the range of 10% of the stride length in each filter iteration. Additionally, the orientation o_i may vary using a Gaussian noise function of 0 and 5 degrees mean and standard deviation, respectively, in relation to the angle obtained by the motion model (see Equation 5.2). Hence, the exact evolution of a particle is defined this way:

$$\begin{aligned} x_i^k &= x_i^{k-1} + (s_i + vs_i)cos(o_i + vo_i) \\ y_i^k &= y_i^{k-1} + (s_i + vs_i)sin(o_i + vo_i) \end{aligned} \tag{5.3}$$

where vs_i and vo_i are the random variables for fluctuation in stride length and orientation, respectively.

Each particle is also assigned a weight factor w_t^m representing its *goodness*. The weights are always non-zero values that add up to one. This value is obtained from an univariate Gaussian distribution with respect to the distance between the assigned location and the nearest wall. That is, the weight of each particle decreases in a monotonic way as its location approximates to any of the walls, as we are assuming that the operator tends to walk keeping a certain distance from them.

After each update, a validation step is conducted in order to verify if some particles followed a trajectory that is not conforming with the map. Because of the randomness of the process, some particles might experience a motion that does not fit with the geometry of the scenario, i.e. crossing a wall. The corresponding particles are accordingly removed from the current set. Each eliminated particle must then be replaced by a new one which is generated using a resampling process from the remaining particles. This resampling process is based on the *Low Variance Sampler Algorithm* [172], and reduces the risk of loosing diversity of the set of particles. The selection involves a sequential stochastic process that gives priority to those particles with a higher weight (w). Each new particle is initialized with the information of the existing selected particle (parent).

The different stages of the particle filtering are iteratively repeated as many times as steps have been detected by the motion model. Figures 5.6(c) and 5.6(d) clearly reflect

the multimodality that typically appears in indoor environments. As it can be observed, this does not suppose a problem to the particle filter which in any case manages to converge to a monomodal case.

Finally, in order to obtain the corrected path followed by the operator, and since each particle stores information about its corresponding parent in the previous iteration, once the final stage has been reached (Figure 5.6(i)), we can perform a backward propagation of each live particle at this stage (Figure 5.6(j)) to obtain the covered path. In this way we can trace the path back to the corresponding starting state, and average the corresponding paths among the M surviving particles.

5.5 Pairwise matching filtering

As we have emphasized during this chapter, one of the most frequent problems we had to deal with while performing the SfM process was the existence of repetitive visual structures in different parts of the scenario. The corresponding visually correct, though positionally incorrect matches, could certainly complicate the construction of the 3D model, a fact that is common practice for typical indoor environments. Because having an error-free reconstructed 3D model is essential for the accuracy of the posterior on-line resection process, in this section we describe how to make use of the previously estimated position and orientation of the training images in order to guide the 3D model reconstruction.

As shown in the previous section, an approximate inference of the location where each picture was taken can be determined through the analysis of the inertial and orientation sensors, given that the operator has followed a pretty predictable pattern of movement. Therefore, since images were time labeled in a synchronized way with the rest of sensor measurements, images can be adequately geo-tagged using the pathway inferred by the particle filter algorithm in consequence. This information can be used to guide the SfM process, thus avoiding the drawbacks like repetitive structures.

More precisely, the method that we have used to build the 3D model in a more unsupervised but still reliable way is based on the generation of a pairwise candidate matching list, that indicates which pairs of images must be compared to look for coincidences between them. In that sense, two images will be matched only if the areas of the scenario

that they cover are fairly well overlapped and they have been taken from similar angles of attack, up to a given threshold.

To illustrate this point, Figure 5.7 (left) shows a pair of cameras whose covered areas significantly overlap. This pair of cameras will be selected to be added to the candidate matching list because both are covering an area where useful and meaningful correspondences can be found. On the contrary, Figure 5.7 (center), shows two cameras that are covering mostly separated areas, thus they will not be included within the matching list. Figure 5.7 (right) shows another useless case in which, despite having a certain area of overlap, the large difference of angles between the optical axis of the cameras (180° in the example) makes it impossible to find good visual matching of the corresponding SIFT features. In fact, a difference between the optical axis of more than 50°, approximately, is acknowledged to clearly degrade the repeatability of features in the original paper describing SIFT [127]. Thus, we will compensate the obtained overlapping area by multiplying it by the following penalty function:

$$f(\alpha_1, \alpha_2) = e^{-(\frac{\min(|\alpha_1-\alpha_2|, 360-|\alpha_1-\alpha_2|)}{50})^{10}} \tag{5.4}$$

where α_1 and α_2 are the angles with respect to the horizontal of each camera respectively. $f(\alpha_1, \alpha_2)$ basically takes a value of 1.0 for angle differences well below 50°, 0.0 for differences clearly greater than that threshold, and smoothly falls from 1.0 to 0.0 for angle differences around the exact threshold (see Figure 5.8).

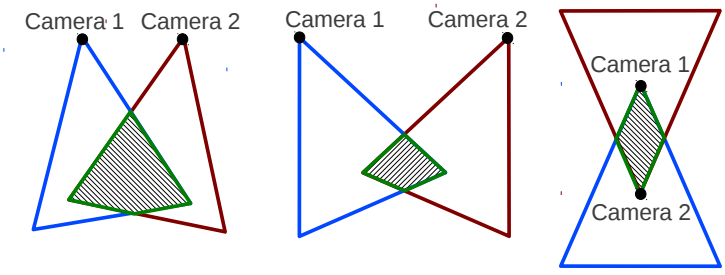


FIGURE 5.7: Overlapping fields of view for different configurations of camera pairs. (left) Pair of cameras whose covered areas significantly overlap. (center) Useless pair of cameras that are covering mostly separated areas. (right) Cameras are not visualizing the same scene.

Figure 5.9 illustrates the process to obtain the area covered by an image according to its position and orientation. On the one hand, from the results obtained after executing the particle filter over the sensor data we were able to estimate the approximate 2D location

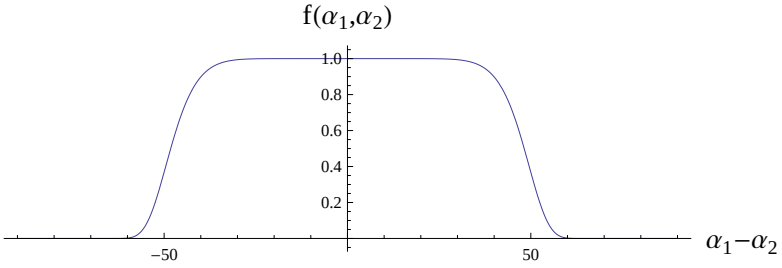


FIGURE 5.8: Graphic representation of the function defined by equation 5.4.

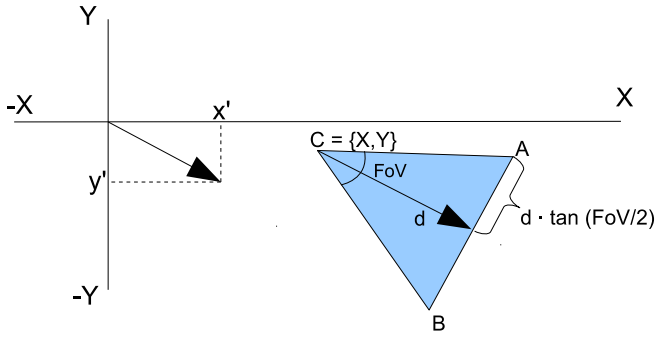


FIGURE 5.9: Approximate area covered by a camera considering the camera center C provided by the particle filtering, the rotation vector that indicated the orientation and the FoV of the camera. The parameter d establishes the maximum distance considered.

in the scenario (X_i, Y_i) at which each image was taken during the training. On the other hand, we also obtained the rotation matrix R that gave us the approximate orientation of the smartphone at that moment using the virtual sensor of the capturing device which adequately combines the gyroscope, compass and gravity sensors. Now, and since the typical camera pose when capturing the images keeps the imaging plane approximately vertical, the unit norm vector $(x', y') = (v_2, v_3)/|(v_2, v_3)|$, with $v = R^\top \cdot (0, 0, -1)\top$, gives us the approximate orientation of the optical axis projected on the floor.

Let $C = (X_i, Y_i)$ be the coordinates of the scenario point where the image was captured (that is, the position of the camera center projected on the floor plane), and d the maximum distance threshold beyond which we discard the obtained SIFT image features. Possible values for d in typical indoor environments will be of a few meters. We have empirically tested different values, obtaining $d = 9m$ gets the pairwise matching list that better performs during the SfM process. Moreover, considering FoV as the horizontal field of view of our camera:

$$FoV = 2 * \arctan 2(camera_{width}/2, f) \tag{5.5}$$

We can estimate the points A and B that define the area covered by a given image as:

$$\begin{aligned} A &= (X,Y) + d(\left(x', y'\right) + tan(FoV/2)\left(y', -x'\right)) \\ B &= (X,Y) + d(\left(x', y'\right) - tan(FoV/2)\left(y', -x'\right)) \end{aligned} \tag{5.6}$$

Using the triangles generated for every camera position, we can compute their corresponding pairwise overlapping areas and compensate them adequately using equation 5.4, as discussed above. The result is a list of matching candidate image pairs that will be used as input for the SfM process. A visual representation of a subset of the cameras used in our experiments is shown in Figure 5.10.

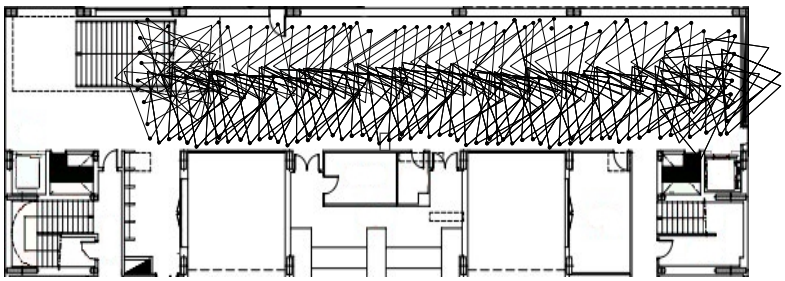


FIGURE 5.10: Overlapping areas example of the images obtained during path illustrated in Figure 5.5(b). (Note that the camera set has been reduced and $d = 5$ in the example for a better image visibility.)

5.6 Experimental evaluation

In this section we evaluate the proposal made in this chapter from the perspective of the reliability of the reconstruction process and its implication over the localization process. There are two different aspects that we have evaluated during our validation:

- We analyze the exactitude of the 3D reconstructions when using the proposed constrained matching process. In relation to this aspect, we check the usefulness of the motion model and particle filtering that determine the position of the cameras in order to guide these reconstructions.
- We also compute the mean estimation error that is obtained using the 3D model when we perform a resection procedure using different input images for testing. These results are compared to those obtained in the previous chapter in order to

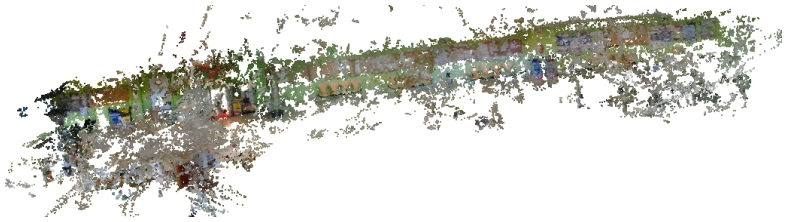
validate the utility of our automatically obtained 3D model when it is used for localization purposes.

5.6.1 3D model generation

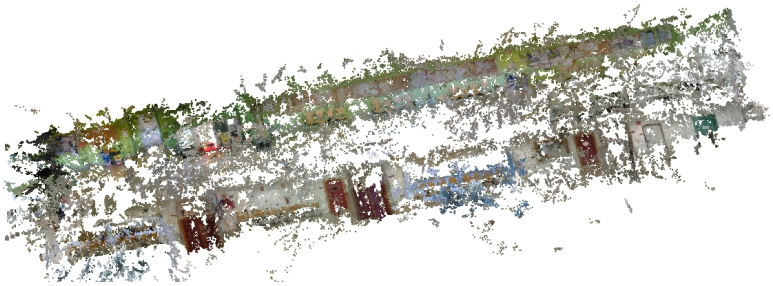
In relation to the generation of the 3D model, Figure 5.11 demonstrates the advantages of using the inertial sensors to guide this task. Figures 5.11(a) and 5.11(c) show two final results of the SfM process (Scenario 1 and Scenario 2 respectively) when the matching is not guided (images were not geo-tagged) and the process is not supervised by an operator. As we can appreciate, false positive matches caused by visually repetitive structures produce models that do not fit well with the corresponding scenarios. In previous versions of our system, these erroneous matchings –extremely frequent in indoor scenarios– had to be removed interactively by an operator in order to get correct reconstructions, in a cumbersome and tedious manual and incremental process.

In contrast, Figures 5.11(b) and 5.11(d), show the correct and precise reconstructions obtained when images were coarsely geo-tagged using the proposal presented in this chapter. Now we avoid the issue of repetitive structures, since distant cameras were early discarded in the pairwise matching list used as input for the SfM process. Apart from the important problem that we solve, the current models were built based on a smaller but also much more reliable set of matches. This implies a significant advantage from the point of view of the matching performance. To illustrate this, Figures 5.12(a) and 5.12(b) show two different *matching matrices* for the 510 images taken from our first scenario. The former is a more dense matrix which corresponds to the situation when a total of 129K pairs of images were initially considered, giving VisualSfM the possibility of the pairwise selection. The later, which used the coarse location obtained by our particle filter, greatly reduces the candidate image pairs, resulting in a much more sparse matrix containing only 30K initial pairs.

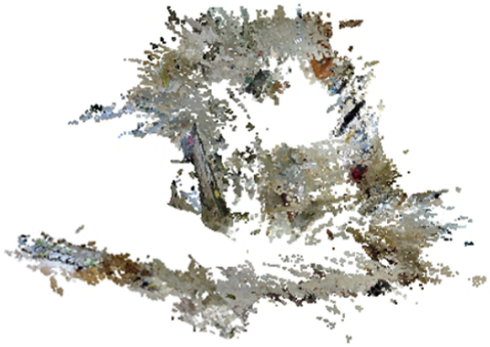
This situation not only involves a great reduction in the computing time of the 3D model, but, what is more important, it also remarkably improves the reliability of the 3D model. In our case, the number of feature descriptors that made up the 3D model in the guided reconstruction is up to 150K instances. In comparison with the 250K feature descriptors used to built the model in Section 4.3.2 of Chapter 4, it might seem that we are losing visual wealth, however as we will demonstrate this is not the case.



(a)



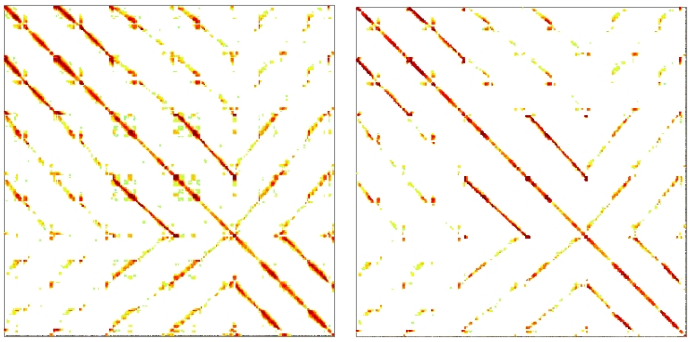
(b)



(c)

(d)

FIGURE 5.11: (a) Final result of the Scenario 1 reconstruction when the matching process is not guided; (b) Final result of the Scenario 1 reconstruction when using our guided matching proposal; (c) Final result of the Scenario 2 reconstruction when the matching process is not guided; (d) Final result of the Scenario 2 reconstruction when using our guided matching proposal.



(a) (b)

FIGURE 5.12: (a) Matching matrix obtained after no guided reconstruction of the Scenario 1; (b) Matching matrix obtained using the guided matching for the reconstruction of the Scenario 1.

Despite the much lower number of features, they represent the real scenario in a more reliable way, since we avoid those cases where false positive matchings could be included in the final model. Moreover, as we prove, localization accuracy is not affected by this circumstance.

Making use of the multisensor approach, we divided the Scenario 1 in four overlapping zones, in the same way as it was described in previous chapter. In consequence, the obtained 3D model was partitioned into four sub-models, each one of them containing around 43k feature descriptors on average. Since we do not expect to carry out a performance evaluation, in order to not introduce more complexity to our analysis, in these experiments we obviated the camera filtering introduced in Section 4.3.5 where nearby cameras were removed to build more relaxing models in terms of number of descriptors. Therefore, considering the four sub-models created, the time required to carry out the matching process was 90 ms on average, a time slightly higher than the one obtained in previous chapter. In consequence, as detailed in Figure 5.13, the total time to perform a entire cycle (from sensor data transmission to 3D estimation) is around 260 milliseconds on average. Similar results were consistently obtained when performing a large number of different reconstructions, which is also relevant since the SfM process involves a considerable randomness in the incremental reconstructions.

Finally, we evaluated *a posteriori* the accuracy of the initial coarse estimations performed by the motion estimator and the particle filtering. Using as ground truth the positions of the cameras assigned by the final refined estimation performed by VisualSfM, it results in a mean error around 2 meters for the whole set of input pictures. As we corroborate

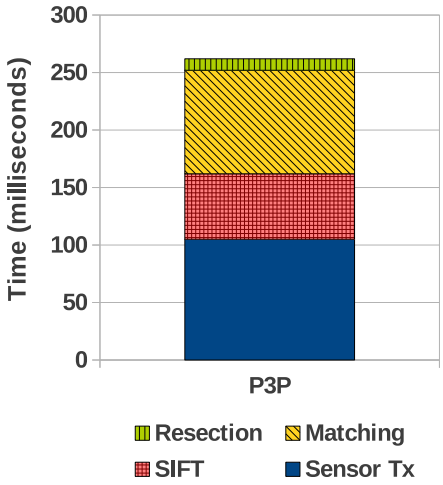


FIGURE 5.13: Time required to complete an entire cycle. It includes the data transmission time, and the time to perform the SIFT features extraction, the matching process (using the multisensor proposal) and the resection process.

with the obtained results, this accuracy is more than enough to correctly guide the matchings carried out during the SfM process.

5.6.2 Localization accuracy evaluation

Once we have demonstrated the usefulness of our multisensor approach to build more accurate and reliable 3D models, we present now some experimental results which demonstrate that we have not only generated accurate 3D representations, but that these are also still fully operational to accomplish the camera resection process that estimates the camera pose and its rotation. Though it was not our initial goal, we prove that localization accuracy has also slightly improved using the automatically built 3D models.

We tested the DLT and P3P resection techniques proposed in Chapter 4 to compare these new results with those obtained previously. DLT is used as candidate for uncalibrated situations whereas P3P is the selected option for precalibrated cases because of its better performance with respect to the Fiores' algorithm. These experiments were accomplished under the same conditions as described in early Section 4.3.5 in terms of the dataset and mobile device involved. The input parameters evaluated for each resection technique correspond with exactly the same that were used in Section 4.3.5.

Table 5.1 and Table 5.2 show the results obtained using DLT and P3P algorithms respectively. Like in the previous chapter, we carried out an exhaustive set of experiments

# Correspondences		40		50		60	
Minimum Inliers		10	30	10	30	10	30
Iterations	Reprojection Error (px)						
100	4	50,1-100%	51,5-87%	54,2-100%	69,1-97%	72,6-100%	67,5-99%
	5	62,2-100%	56,2-89%	83,7-100%	68,6-100%	65,6-100%	67,8-99%
	6	62,2-100%	60,3-91%	159,9-100%	80,6-100%	108,2-100%	72,4-99%
200	4	50-100%	46-87%	56,4-100%	69,2-97%	55,8-100%	58,7-100%
	5	55,5-100%	68,8-91%	59,2-100%	70,1-100%	66,9-100%	68,3-100%
	6	51,7-100%	62,4-91%	77,6-100%	139,1-100%	76-100%	59,4-100%
300	4	45,8-100%	45,5-87%	53,2-100%	74,8-97%	57,9-100%	58,8-100%
	5	52,4-100%	62,9-91%	64,3-100%	83,7-100%	56,6-100%	67,7-100%
	6	58,7-100%	58,3-91%	75,8-100%	78-100%	65-100%	64,1-100%

TABLE 5.1: Results of the resection process using DLT and different configuration parameters. Estimation Error (cm) - Resection camera successful cases (%). White shading cell represent the most accurate option. Red background cell indicates the configuration that gets the best trade-off between accuracy, performance and reliability.

Misalignment Error	0,01			0,03			0,05		
Minimum Inliers	10	20	30	10	20	30	10	20	30
Residual Value									
0,0001	9,1-78%	7,9-79%	7,7-76%	9-49%	8,1-55%	8,7-51%	8,2-35%	7,9-39%	8,1-41%
0,0005	4,5-100%	3,6-97%	3,6-92%	8,4-99%	7,4-97%	7,7-92%	11,4-98%	11,2-95%	11,5-92%
0,001	4,9-100%	3,6-97%	3,6-92%	8,4-99%	8,5-97%	7,5-92%	13,2-97%	13,7-96%	13,8-91%
0,0015	5,3-100%	3,5-98%	3,5-92%	8,9-100%	8,9-98%	9,4-92%	15,9-98%	13,6-97%	13,5-92%
0,002	9,7-100%	4,1-100%	3,9-96%	12,2-100%	15,4-96%	18,1-96%	44,1-94%	17,6-95%	17,8-95%

TABLE 5.2: Results of the resection process using P3P algorithm and different configuration parameters. Estimation Error (cm) - Resection camera successful cases (%). White shading cell represent the most accurate option. Red background cell indicates the configuration that gets the best trade-off between accuracy, performance and reliability.

in order to assess the performance (accuracy and feasibility) of different sets of configuration parameters for these algorithms. We executed each algorithm 20 times per image and per configuration. That is, the shown values are the averaged results of a total of 400 executions. It is worth mentioning that these tables only show part of the whole set of tested configurations. Performance evaluation was previously analyzed in Chapter 4.

The configuration parameters (red background cells) which offered the best trade-off between accuracy, performance and reliability, have minimally varied from those selected in the previous chapter. These variations are understandable due to the differences between the 3D models used in both experiments. In the case of DLT algorithm, since we were using more precise 3D reconstructions, fewer iterations of RANSAC were needed to obtain a good accuracy, what implied a noticeable time reduction of the resection process. For that same reason, in the case of P3P, we could be more restrictive with respect to the misalignment error required to select the inliers during the RANSAC process, what has a direct impact over the final accuracy obtained.

Regardless of the influence of the random RANSAC process, it is possible to confirm

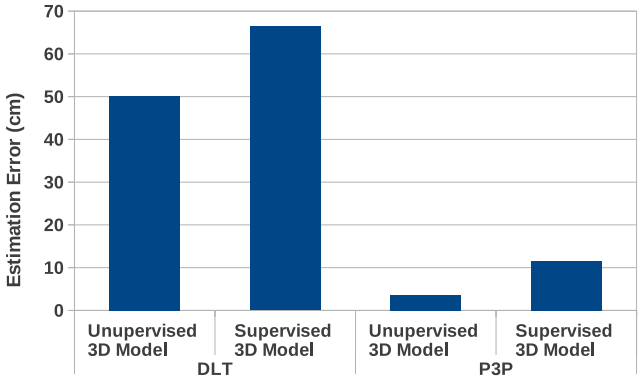


FIGURE 5.14: Accuracy obtained using DLT and P3P resection techniques comparing the utilization of unsupervised and supervised built 3D models.

the improvement in terms of accuracy compared to the results shown in Section 4.3.5. Figure 5.14 shows a comparative analysis between the currently obtained results and those obtained in the previous chapter. In those cases where the camera autocalibration was performed, the estimation error has been reduced down to 50 cm, keeping the same reliability close to 100% of executions. Moreover, in calibrated cases, the accuracy obtained reaches the 4,1 cm of estimation error while the reliability of our proposal still maintains over 100% of tested cases, which ensures a good system performance.

We want to emphasize that these results confirm the importance of this contribution in order to build accurate 3D models in an unsupervised way. We have not only proven the accuracy of the obtained reconstructions but also have demonstrated their usefulness to provide a precise estimation of the devices position. Apart from the accuracy obtained when estimating the camera center, another important factor is the accuracy of the estimation of the camera rotation. Therefore, though we did not evaluate the estimated camera rotations in a quantitative way, we performed an exhaustive qualitative assessment. For that purpose, we carried out a set of tests making use of the real-time augmented reality prototype described in following section.

5.6.3 Real-time augmented reality prototype

As stated in the previous experimental section, the response time of our resection service by the server is typically limited to less than 260 ms. This refreshment rate is still insufficient to fully cope with video input rates of mobile devices (25 frames per second in typical cameras). Though we can think of many mobile applications that do not

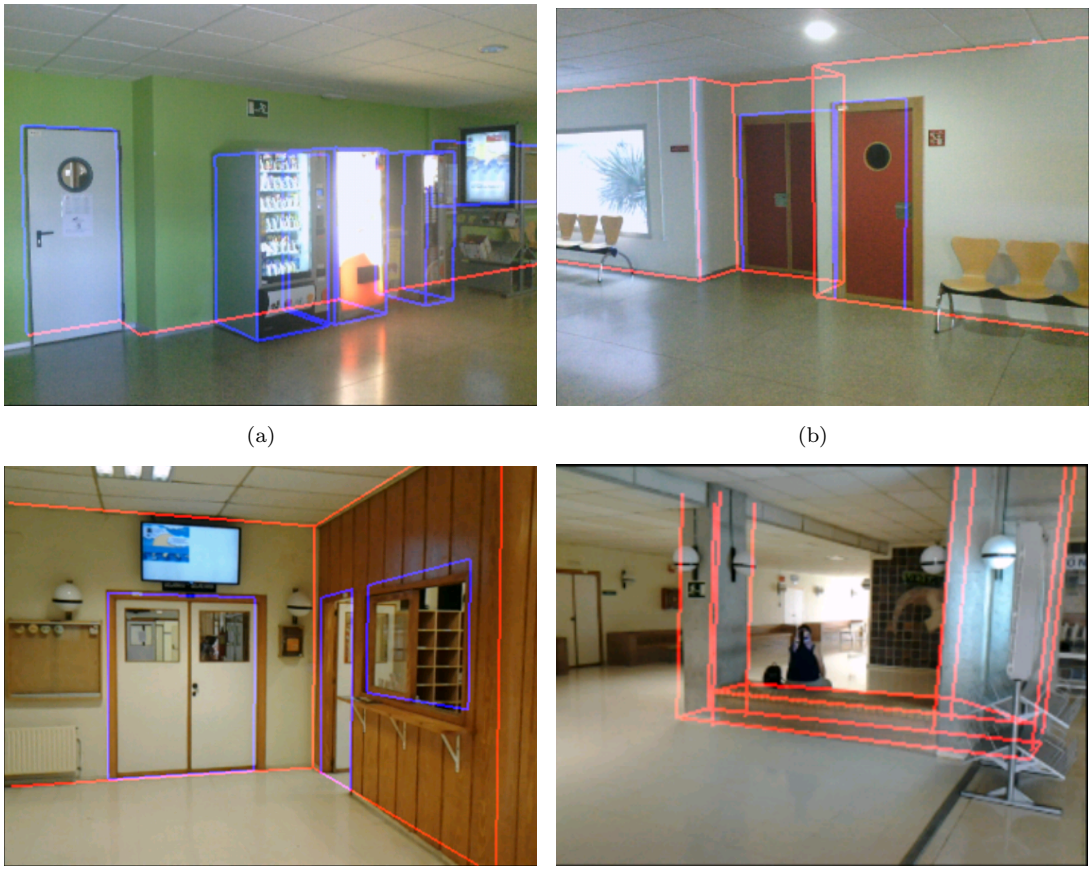
need freshly updated 3D position and orientation of the device at video input frame rate, many other applications would certainly benefit from a location service providing continuous, always on, real time localization. For example, an application providing information on where your friends are inside a particular building could be just based on a 2D map showing mobile icons referring to your friends, but it would be very useful if it were also able to display continuously updated augmented reality information on top of the video input, captured by the camera device as the user keeps on moving. Users could even switch from one interface to another, depending on the kind of information they are interested at that moment.

Moreover, the robustness and overall behavior of the application could certainly suffer if the location system tries to update the location information by making continuous queries to the server, due to both the communication overhead and the eventually unavoidable resectioning failures. To avoid such degradation of the user experience, while allowing also a smooth and real time image refreshment rate, we make use of some additional sensors provided by the device, such as the accelerometer and the gyroscope, which allow us to detect different types of motion and reduce the needed communication with the server. This is another clear advantage of the multisensor integration.

In order to test this hybrid approach, we developed a prototype application for Android devices based on OpenGL ES 2.0 which is able to display augmented reality based on precise estimations of the full 6 degrees of freedom: (X, Y, Z) position and (α, β, γ) Euler angles for the rotation matrix. We display a wire-frame model on top of the real world to visually check the accuracy of our estimations. Figure 5.15 shows some snapshots of the prototype working in different environments.

Every time a new rotation of the device is read from the sensors (typically at 50 Hz rate in most Android devices), the original rotation matrix estimated by the last successfully resectioned image is modified accordingly, and the augmented reality is updated. The camera center C is not modified at all. Only when a translation movement is detected by the accelerometer the device is forced to request a fully new estimation of position and rotation from the server. The continuous update of the current device rotation is performed using the gyroscope sensor according to the following formula:

$$R_{new}(t_{cur}) = R_{comp} \cdot R_{gyr}(t_{cur})^{\top} \cdot R_{gyr}(t_0) \cdot R_{comp} \cdot R_{cam}(t_0) \tag{5.7}$$



(a) (b)

(c) (d)

FIGURE 5.15: AR results: a-b) Scenario 1. c-d) Scenario 2.

Here, $R_{cam}(t_0)$ stands for the last rotation estimation provided by the server using strictly visual information, and is therefore only refreshed when the user clearly moves to a new location, as stated above. $R_{new}(t)$, on the contrary, is updated at the systems sensor reading rate, using the most current gyroscope information. This provides continuous, smooth visual feedback to the user. The update is based on the fact that the matrix product $R_{gyr}(t_{cur})^\top \cdot R_{gyr}(t_0)$, with $R_{gyr}(t)$ the absolute rotation read by the gyroscope of the device at time t, gives at every moment the *relative motion* between the moment the picture corresponding to the last camera resection was taken ($t = t_0$) and the current instant of time ($t = t_{cur}$), assuming that the movement performed by the user is a pure rotation (i.e., there is no physical translation of the camera center). Finally, the matrix $R_{comp} = diag(1, -1, -1)$ is a simple diagonal matrix that acts just to compensate the different *handedness* of the camera and world systems. This matrix is useful when using tablets devices, since they are expected to be used in landscape orientation.

Given this freshly updated rotation, the final projection matrix which is used at every moment to perform the projection of virtual structures for augmented reality is $\mathtt{P} = \mathtt{KR}_{new}(t_{cur})[\mathtt{I}| - C(t_0)]$, with the value of the camera position $C(t_0)$ as estimated in the last query to the resection service in the server, and the rotation matrix $\mathtt{R}$ substituted by $R_{new}(t_{cur})$, computed as stated in equation (5.7). Note that this is a very fast operation that only changes in the most current $R_{gyr}(t_{cur})$ reading, and can thus be performed at a very high refreshment rate even in low-end mobile devices. As we tested during the experiments, again, the multisensor integration allowed to improve the user experience, providing a comfortable visual refreshment rate.

5.7 Summary

Throughout this chapter we have presented a novel method to robustly build visual 3D models in a mostly unsupervised way. This robustness is accomplished by integrating the data acquired by the inertial sensors of the training device, usually a smartphone or tablet. Making use of motion estimation techniques and integrating a particle filter we generated a set of geo-tagged images used to constrain the matching process involved in the SfM procedure. Our main contributions were:

- We designed an effective technique to avoid one of the key problems in large SfM reconstructions, the repeatability of objects in scenes.
- We introduced the use of a constrained pairwise matching list that increases the robustness of the SfM process. This filtering process represents an important contribution to favor the system scalability. The obtained 3D reconstruction result in a more lightweight model, which reduces the computational load of the matching process during the on-line phase.
- We have validated the benefit of our proposal demonstrating the usefulness of the obtained reconstructions to provide accurate 3D location estimations.

According to the results obtained, we can confirm that we have achieved a good trade-off between the robustness of the 3D reconstruction and the accuracy (around 4 cm of averaged error), the reliability (100% of successful estimations) and the response time

reached (around 260 milliseconds) during the on-line phase. We developed an augmented reality application for Android devices based on the described localization service, which relies also on the information provided by the gyroscope in order to adapt the overlaid information to the device's rotations in real time. The integration of the gyroscope within this application allowed us to minimize the delay during the localization step, thus improving the user experience.

Finally, we are aware that our solution may suffer from scalability problems when dealing with large environments, like malls, airports, etc. There are severals key points that should be addressed in this concern, as for example, the path to be covered by the operators during the training phase, in order to establish certain spatial restrictions that support the particle filter process. The motion pattern defined for the training phase might increase the time notably when we try to model such large scenarios. A future multidevice integration, using more sophisticated pedometers, might allow to define a usual walking pattern to accomplish the training tasks. To conclude, we consider our proposal a valuable initial step that solves some of the previously described issues and opens new ways to continue the evolution of this kind of solutions.

Chapter 6

System architecture

This chapter describes the architecture which encompasses the overall localization system that has been proposed throughout this dissertation. We give a detailed description of the different modules which are part of this architecture. Furthermore, we introduce the communication interfaces that facilitate the flow of information.

As we commented in the introduction of this thesis, one of our main aims was to design a distributed, rather than monolithic, architecture supporting the design and development of a location service able to combine different type of sensors. We will demonstrate that our proposal can be used independently of the type of sensors managed, their specific characteristics and the processing resources required. Among its strengths we can stand out its adaptability to different configurations. This capacity is achieved by the specification of well-designed entities or modules that can be placed at different nodes which can be treated as logical entities instead of as physical equipment.

This architecture is structured in different layers, in the way other works like the *location stack* [83] or *Loc8* [165] do, separating the inherent functionality that must be integrated within this type of solutions. Our initial proposal is based on a client-assisted model that suggests a division of responsibilities between the mobile devices and dedicated servers (system core) according to the imposed requirements and the available computational resources. Though the reading of previous chapters is detailed enough to understand the system structure and to identify the different tasks performed by either the client-side or the server-side, now we provide additional information which helps the reader to obtain

a clearer perspective of the workload distribution and the workflow among the defined components.

Our proposal is technology independent in several ways:

- It offers a high level interface that supports the development of different external location-based services independently of the technology selected to carry out their implementation. A high level API has been implemented following the *representational state transfer* (REST) [150] architectural style over HTTP connections, providing the necessary functionality to interact with our core system, e.g. to obtain positioning information.
- Its design minimizes the coupling among the different technologies available for localization. Different localization algorithms can be developed independently of the types of sensors considered. All of them can be integrated within the same system, using the most appropriate one at each situation, depending on different factors, such as the type of sensors available or the accuracy required.

For these reasons, the design of this architecture tries to fulfill the following statements:

- **Extensibility**: The defined schema is open for the future integration of additional technologies and localization algorithms, in order to cover the necessities and opportunities that could appear from the integration of additional sensors in future generations of mobile devices. In the same way, new localization algorithms can be added to make use of new context information or to improve accuracy.
- **Adaptability**: The system must be able to provide the most appropriate services according to the characteristics of each particular environment. This includes the use of the available sensors and the possibility of supporting different location-based query types (like positioning, navigation or range), providing different accuracy levels according to the requirements imposed.
- **Modularity**: Our architecture must be modular enough to facilitate the distribution of the functionality implemented among the different devices that take part in the system, according to their computational resources, connectivity, etc.

- **Scalability**: Every location-aware system should be able to offer a good quality of service in terms of accuracy and response time, independently of the scenario of deployment, its dimensions and the number of users that make use of the service simultaneously. Though we do not provide specific solutions to address the scalability issue, the decoupled and modular nature of this architecture favors the design of scalable solutions in a most appropriate way than a centralized approach.

Taking into account these premises, in this chapter we provide a detailed description of the architecture proposed. Although we have fully implemented our architecture in order to perform the experiments presented in previous chapters, we will avoid the specification of implementation details; rather, our main goal will be to provide a clear description of the system organization, leaving the responsibilities of the implementation decisions to developers.

Our architecture is agnostic from the point of view of the security. This is an important issue which can be addressed from different perspectives, for example, setting up some security measures to get access to sensitive data for reading or modification, or establishing encryption mechanisms that ensure the privacy of the users. Considering that the architecture we propose is mainly focused on the localization perspective, the security aspect has not been addressed within this work. Nevertheless, we are aware of the importance of addressing this shortcoming in the future.

6.1 Architecture overview

The presented architecture is based on previous works already introduced in Chapter 2. Influenced by the design of those previous approaches, as we show in Figure 6.1, we have designed a layered stack in which we can clearly distinguish three main components, the *context model*, the *space model* and the *location engine stack* itself. We are going to describe each one of the layers and how we have divided their functionality in different views.

6.1.1 Entities definition

The architecture shown in Figure 6.1 consists of several entities or layers representing the different abstraction levels. These entities are:

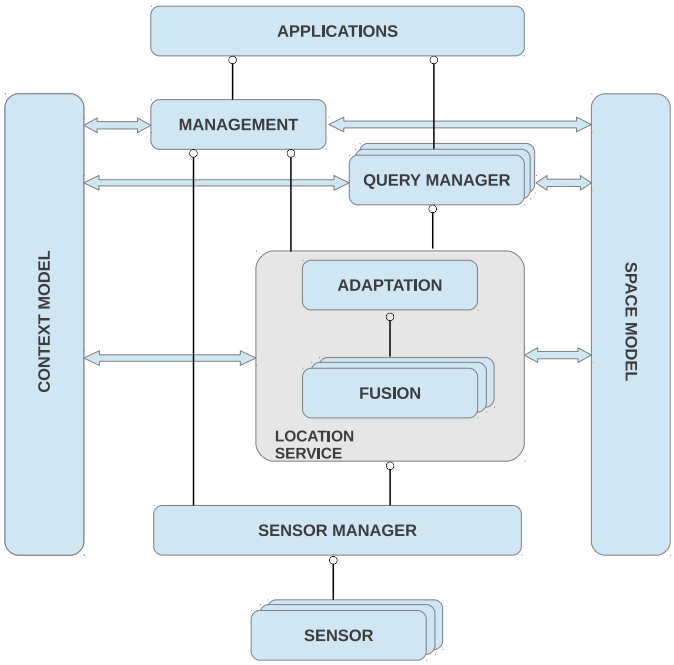


FIGURE 6.1: System architecture overview.

- **Context model**: This entity defines each one of the data elements that take part of the location process. On the one hand, it contains information about the components involved in the localization process. For example, information of the different APs that make up the WiFi network (like their MAC addresses, or their position, if known), or data about the devices that will be the object of positioning (device's identification, calibration information of their sensors). On the other hand, it is the responsible for the management of the sensor information collected during the training and the on-line phases. Therefore, it handles the information regarding the models, e.g. WiFi and images fingerprint maps and 3D models. All this information is available to be accessed by the rest of components of the architecture by means of an API that has been defined to that effect.

- **Space model**: The information related to the description of the physical distribution of the environment (zones, rooms or cells), their relations of adjacency and hierarchy, and the support of different maps, is handled by the *space model*. That

information will be accessible by means of the correspondening API provided by this layer.

- **Application**: This is the highest abstraction level layer of the system architecture, and it represents the applications that make use of the services provided by our system. Furthermore it also includes those other applications that have been developed in order to participate in management and deployment tasks. This layer will be able to access to the functionality provided by the *management* layer when it is required. Throughout this chapter we will refer to the *application* layer to identify these two different types of applications, the external applications that make use of the location services (e.g. augmented reality applications), and the management ones that are used to carry out the system deployment (e.g. training applications).

- **Management**: This layer has been designed to support the management and deployment tasks. It includes methods to accomplish the process of sensor scan, and a location API which is used during the training phase when the models are still not available. External applications are able to use some of these services, for example to carry out the sensor scan, exempting of this responsibility to the *application* layer. This layer is able to access the *context model* and *space model* data using the APIs they provide.

- **Query Manager**: This entity processes the queries performed by external applications and returns the appropriate response according to the information provided by the *location service*. Our current system supports three different types of queries: localization, navigation and range queries. External applications will be able to request these type of services, as we describe in Section 6.6. This layer interacts with other elements of the architecture in order to solve the different types of queries supported. For example, it can communicate with the *space model* to solve simple navigation queries or it can access to the information stored in the *context model* repository to reply to range queries. Different instances of the *query manager* layer can be defined in order to handle different types of queries.

- **Adaptation**: This is the highest layer of the *location service*. It is in charge of analyzing the requirements indicated by the *query manager* in order to invoke the most appropriate instance of the *fusion* layer. Depending on several factors (the

accuracy required, the scenario of application or the sensors availability, etc.) it balances the received requests among the *fusion* layers available. The definition of this entity is a key point to offer an adaptable service and to favor the extensibility of the system. Once the request has been solved by the *fusion* layer, it will be adapted to the requirements expected by the application. That is, sometimes the system is able to obtain a position with a better accuracy than the one requested by the services. In these cases, the response must be adapted to the required accuracy level. Finally, this entity is in charge of managing cached information. Therefore, if the position of a device at a specific moment is required and it was previously estimated, the *adaptation* layer will return the cached result, instead of computing the position again.

- **Fusion**: This module uses the sensor data in order to produce a time-stamped representation of the geographical positions of the mobile devices. Depending on the required accuracy level and the type of measurements, this entity makes the most convenient utilization of the available algorithms. It is possible to include several *fusion* instances within the same *location service*, each one of them managing different localization algorithms for one or more space models.

- **Sensor manager**: The measurements obtained from each sensor are filtered and processed by this entity in order to adjust RSSIs values depending on the calibration parameters of a specific device, to discard blurry or uniform images that are not useful for further processing, to extract image features, or to filter noisy values from the accelerometer, among other functions. The functionality related to the *sensor manager* can be placed on dedicated servers or can be directly performed by the mobile devices, depending on their computational resources and the requirements to accomplish the specific task. Once the measurements have been processed, the resulting information populates the *context model*, making them available to be accessed by the *fusion* layer in a posterior localization stage. At this point we want to remember that our architecture is able to carry out the sensor data acquisition process and localization estimation process as independent tasks. That is, sensor measurements can be collected without the need of being required for an immediate localization, and vice versa, locations estimations can be performed using previously collected measurements.

- **Sensor**: It is the lowest abstraction level of the architecture. The *sensor* layer represents the hardware and drivers for capturing the new measurements from the sensors available at the mobile devices. The functionality offered by this layer depends on the APIs provided by the corresponding OS.

6.1.2 Division of functionality in views

The functionality implemented by each one of the different layers already introduced has been structured in three different perspectives or views. This division has been accomplished with the main aim of providing a better organization of the functionality and responsibilities of each layer that takes part in the architecture. We refer to a view as a way of grouping those tasks which have a strong interrelation among them. The defined views are:

- **Sensor view**: It includes all the functionality related to the collection of contextual information using the sensors available in the smartphones. It defines the tasks performed during the training phase in order to obtain a collection of measurements that will be used to build the system model. In addition, this view includes the definition of the process that takes place during the on-line phase, where the obtained information is used to estimate the position of the device.

- **Management view**: It defines those tasks regarding the generation of the sensor models and the system maintenance. Using the sensor data obtained during the training phase, this view involves those processes in charge of creating the fingerprinting maps based on WiFi signals or images, generating the WiFi-based probability distribution models, the 3D models based on images features or the estimation of the path covered by the operator during the training using the inertial measurements, among others. Furthermore, additional tasks as the calibration of sensors are specified as management processes as well. Finally, though not included in the current version of our system, those tasks related to the organic generation of fingerprinting maps (using crowdsensing techniques in the way they are presented in [97, 142]) would also be included within this view.

- **Location view**: This view encompasses the functionality required to solve the different queries made by the applications. It includes the different algorithms and

techniques which have been described in previous chapters regarding the utilization of WiFi RSSIs and images to estimate the position of the devices. The algorithms in charge of solving the navigation and range queries, are also included as part of this view.

There are some situations in which the different views are interconnected since the functionality included in any of them is required to complete an action which is defined in other view. That could be the case of the process to generate the sensor models, which even being a task included in the *management view*, requires the functionality defined in the *sensor view* in order to carry out the sensor data acquisition.

6.2 Context model

Once we have a clear perspective of the structure of the proposed architecture, in this section we describe the main characteristics of the entity defined as the *context model*. As we have clearly stated throughout this thesis, location services are based on the extraction and analysis of environmental information. Consequently the *context model* defines the appropriate structure to handle this information and provides the mechanisms that ensures its availability. As the rest of entities of our architecture, the context model has evolved during the development of this thesis. In this case we have taken into consideration some interesting aspects proposed in works like those made by Stevenson et al. [165], Martinez et al. [128] and Beckkelien and Deriaz [32], but addressing our design to provide a global solution.

The *context model* is in charge of managing the measurements collected by the different sensors that remain under the control of the *sensor manager*, both during the training and the on-line phases. We defined the concept of *observation* as the element that integrates the sensor information collected by one device in a specific instant of time. According to our current specification, each *observation* may include the data obtained from the following sensors:

- *WiFi*: this measurement consists of a list of the RSSIs observed from every available AP. For convenience, this information is normalized to the value of percentage

of reception, converting the RSSI in *dBm* to *0-100* values. These values are calibrated to make them compatible with those used to build the fingerprint maps. We save both the raw and the processed data into the repository.

- *Image*: contains a binary object that represents the image captured by the device's camera. Images are captured in *jpg* format with a *640×480* resolution. We apply a compress ratio to minimize the transmission overload, but without affecting the system accuracy. Once the image is received at server side, image features are extracted and stored in the repository. However, in case of using a mobile device able to extract the features from the images, these features could be transmitted instead as a list of vectors of numerical values. The vector size would depend on the type of descriptor used.

- *Accelerometer*: includes a list of measurements of the acceleration force that is applied to a device on all three physical axes (X, Y, Z). Due to the usual high refresh rate of this sensor (50Hz), an observation may contain raw data but also aggregated statistical information from several measurements, e.g. mean or standard deviation values. In our case we also process these measurements to get a coarse estimation that indicates whether the device is still or in motion.

- *Digital Compass*: this sensor indicates the orientation of the device, represented as the deviation angle between the X axis of the device and the magnetic north, which is the value we save into the repository.

- *Rotation Vector*: this is a virtual sensor present in most of the mobile devices. It combines the information from the accelerometer, the gyroscope and the magnetometer, building a 3×3 matrix which represents the angle of rotation over the three axis (X, Y, Z). This matrix is saved in the repository as a list of numerical values.

It is worth mentioning that it is possible to extend the *observation* element including other sensor information, like GPS measurements, sounds from the microphone, etc., in case it would be necessary to support other types of services or to improve the accuracy of the solution presented. The way in which we store the data is just a suggestion, it can be adapted to the specific requirements of the system to design.

Apart from managing sensor information, the *context model* is also the responsible of handling all the information regarding the infrastructure elements. It includes information of the mobile devices registered in the system, such as its identification data (e.g. the MAC address of their WLAN interface) and the device model, which is useful to associate each device with specific sensor behavior. For example, the information about the WiFi transmission range that enables the calibration of this sensor with respect to a reference one, or the focal length of the camera which allows the use of more accurate techniques to obtain the 3D location of the device, can be associated to the device model.

The information about the different models that support the localization process is also included as part of the *context model*. Within this category we differentiate the following data sets:

- WiFi fingerprint maps used by deterministic methodologies, probabilistic models, or models based on the relative order of the signals strength.
- The information related to the cluster-based division. This information supports the definition of the different zones in which we can divide the environment according to the signals characterization.
- Information regarding the calibration process of the WiFi sensor. According to the technique that we use, this information refers to the reference points that should be visited by users to calibrate this sensor in case that no similar devices were previously calibrated.
- Transition models that support the inference of people movement throughout the scenario. This mainly refers to the transition matrices required to integrate the HMM within our system model.
- Fingerprint maps of images. They are made up by the set of descriptors associated to the features extracted from collected images. This information was used for the implementation of place recognition based models.
- Image-based 3D models that represent the environment. In this case, they support the implementation of the solution that uses camera resection techniques to obtain a precise estimation of the 3D position and orientation of the devices.

Finally it is worth mentioning that all the information managed by the *context model* is accessible by the rest of layers by means of a RESTful HTTP API whenever they are not implemented in the same physical device. Otherwise they can invoke the functionality provided by the API of each layer as local services. We will provide more information about the communication with the *context model* in following sections.

6.3 Space model

As it was discussed in Chapter 2, there are some previous works which addressed the design of the spatial model providing a set of primitives that allow the description of regions of space and the relationships among them, like those presented by Ye et al. [190] and Becker et al. [30], among others. As a mixture of the characteristics offered by these previous works, our *space model* (SM) provides the necessary mechanisms to manage the concepts of hierarchy, adjacency, containment, disjointness, overlapping and connectedness among the different spaces that made up the scenario of interest. Here we only define part of the design of the SM, as the full design of a spatial model is complex enough to be treated in a specific thesis. Our main intention is to facilitate the understanding of this work.

Within our design, we define the concept of SM as the higher abstraction level that represents the global scenario where the location service is deployed. It may represent just a building (i.e. a university faculty), or a building complex (i.e. a university campus made up by different buildings). We also define the concept of *space model element* (SME) which refers to each region of space that is part of the SM. The definition of these entities makes this approach extensible and scalable, since they allow us to work at different abstraction levels, providing the ability to model any type of scenario where our solution is deployed.

From the point of view of their hierarchy, Figure 6.2 represents the different types of SME defined, each one of them representing different granularity levels that provide the necessary tools to represent a scenario, independently of its shape and physical distribution. The different levels defined in this hierarchy are:

- *Point or cell*: Corresponds to the centroid of a specific cell defined on the scenario. Each point will be determined by a 3D coordinate (x, y, z) that establishes its

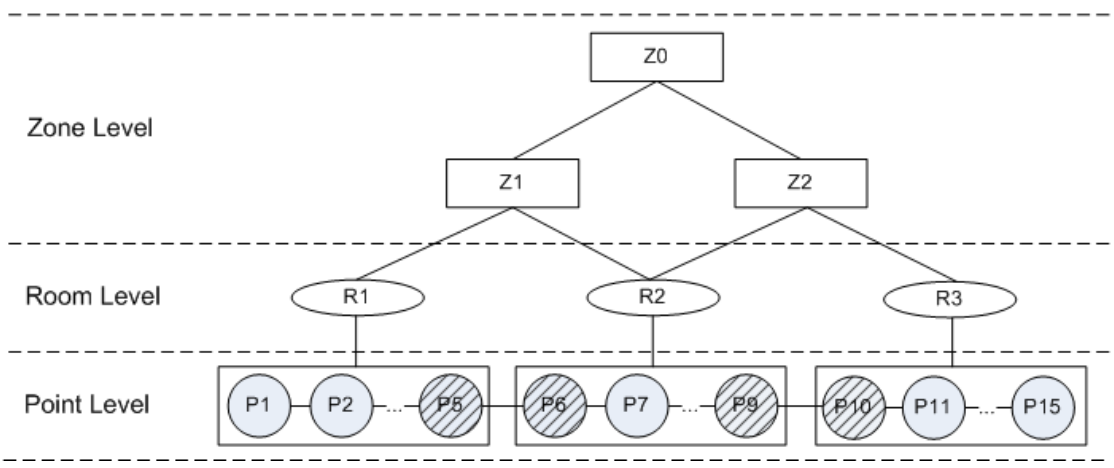


FIGURE 6.2: Hierarchical and adjacency diagram of the *space model*. Weaved points represent the *entry points* that define the adjacency between rooms.

position. It is defined from a geometric point of view, and represents the lowest level within the defined hierarchy. This is the higher level of accuracy that can be obtained by fingerprint-based localization techniques.

- *Room*: Consists of a group of points that represents a closed area, room, office or lab, but also open areas, corridors, stairs or landings, among others. These elements have been defined from a semantic point of view (e.g. the library) but we also support the definition of the polygons that represent their geometry.

- *Zone*: Refers to a larger area that is made up by several rooms. Additionally, a zone can be formed in turn by other zones, for example, each floor of a building can be considered as a different zone, which at the same time is composed by different smaller zones that group several rooms.

Though not defined as part of the space model, our location service is able to provide accuracy finest than point level. These estimations represent a 3D position including the orientation of the device within the coordinate system defined by each space model. Then considering the geographical situation, we are able to associate them with higher abstractions level positions (point, room or zone).

The adjacency among SME was modeled at different abstraction levels. As we can see in Figure 6.2, the concept of adjacency was defined at *point* level. In that sense *point P1* and *point P2* are considered adjacent since *P2* is directly reachable from *P1*. The proposed design supports unidirectional and bidirectional adjacencies, which allows the representation of one way transitions. To manage the adjacency at *room* and *zone* levels,

it was necessary to define the concept of *entry point* (EP), already introduced by Hu et al. in [89]. One EP is defined as a *point* element whose coordinates are close to the entry or exit of a *room* or *zone*. In that sense, two rooms *R1* and *R2* will be considered as adjacent whether one of them contains, at least, an EP which is adjacent to an EP defined in the other. That is the case of points *P5* and *P6*, which establish a relation of adjacency between *R1* and *R2*. At the same time, since *R1* belongs to *Z1* and *R2* belongs to *Z2* we can consider both zones as adjacent too.

Once the adjacency between spatial elements has been defined, the *space model* is able to support navigation queries in order to determine the path that connects two different regions of the scene. Additionally, and supported by the defined hierarchy, these navigation queries can be solved at different levels of abstraction, when room or zone level is required. It is important to mention that the current design does not support the connection between different space models, since it has not been integrated within any outdoor *geographical information system* (GIS).

6.4 Sensor view

As we introduced previously we have distributed all the functionality within three different views. In this section we introduce the first of them, the *sensor view*, which specifies the interactions among the architecture components involved in the sensor data collection and the transmission of the data collected to the repository managed by the *context model*.

The main components involved in these processes are the *sensor* and *sensor manager* layers. The functionality provided by the *sensor* layer is accessible by means of the API provided by the OS installed at the device. Meanwhile, the processing tasks performed by the *sensor manager* have the main goal of controlling the data acquisition and analyzing the raw data obtained in order to:

- Control the available sensors and obtain information from them according to the accuracy required by the final application. For example, in case of supporting AR applications, WiFi RSSIs, images and gyroscope readings are required to be measured. On the contrary, an application just requiring proximity only needs to scan the WiFi networks.

- Process the obtained measurements to infer coarse-grained estimations, e.g. to calculate coarse-grained motion estimations from accelerometer information.
- Carry out an admission control procedure to avoid sending those raw data that for any reason are not useful for localization purposes, e.g. blurry or uniform images.
- Minimize the amount of data transmitted to reduce the communication overhead. For example, performing the extraction of image features and transmitting them instead of image itself, in case of being able to perform this process in the mobile device. In the case of inertial sensors, coarse estimations or statistical calculations (motion detection in the case of the accelerometer) can be sent instead of raw data.

The functionality included within this *sensor view* regarding the data acquisition is used during the training and the on-line phases, supporting those tasks involved in the rest of views. The *sensor manager* is in charge of transmitting the collected measurements to the repository managed by the *context model* making use of the API that it provides. We have defined different transmission policies to indicate the *sensor manager* the most appropriate frequency to send the available measurements. These policies are:

1. As soon as a new observation (made up by new measurements from all sensors) is obtained.
2. When a new measurement from one specific sensor is obtained.
3. At a specific rate, using the most recent measurements from each sensor.

The choice of one of these policies will depend on the scanning process implemented by the OS and the localization requirements. For example, in case of running an AR application, it would be necessary to send a new observation as soon as a new image is collected (2^{nd} policy). Cached information of other slower sensors can be also used, e.g. the last obtained WiFi RSSIs.

Figure 6.3 shows the sequence diagram that represents the different steps which take place during the process of sensor scanning in the on-line phase. In this example WiFi RSSIs, images and inertial measurements are obtained and stored into the repository managed by the *context model*. The process starts when an application, which is installed at the mobile devices, initiates the data collection using the API provided by the

management layer (step 1 in Figure 6.3). Then, the *management* layer communicates with the *sensor manager* (step 2), which starts the data acquisition using the required sensors (steps 3 and 5). Each of the sensors is initiated in a different thread. Next an iterative process (steps 6 to 9) starts. When a new measurement is obtained from one of the observed sensors, this fact is notified to the *sensor manager* which processes the data received (e.g. to discard blurry images). When the *sensor manager* is able to build a new *observation*, it is saved into the repository (step 9). This process is influenced by the transmission policy selected. In this example, the sensor data are transmitted when at least one new measurement of each sensor is obtained. Finally, the data acquisition ends when the *application* considers that no more positioning information is needed (step 10). Then the *management* layer orders the *sensor manager* to stop the scan (step 11), which in turn sends the notification to the sensors (steps 12 to 14). The functionality associated with the *sensor manager* is balanced between the smartphone and the server. For example, due to the difficulties to extract the features from images at the mobile device, we implemented this method in the server side. Nevertheless, if possible, it can be done by the mobile device itself.

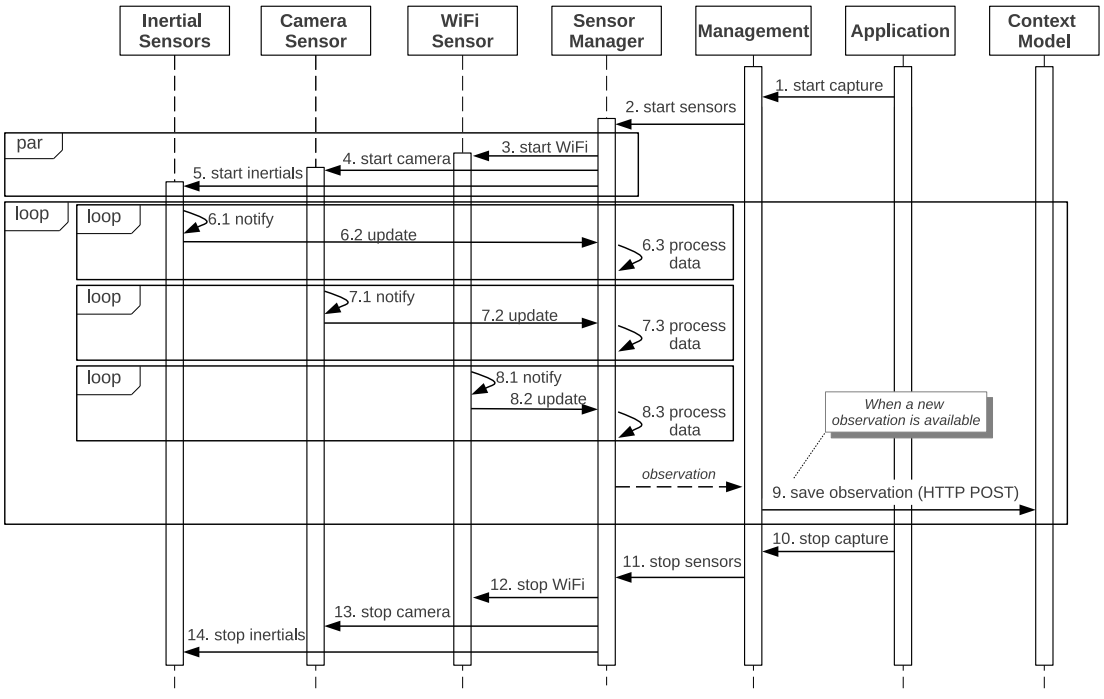


FIGURE 6.3: Example of sensor data acquisition during the on-line phase.

Although we focused on the on-line phase to illustrate the flow of information that takes place at this *sensor view*, we want to emphasize that a similar process is performed during the training stage. The main difference relies on the fact that the obtained

measurements must be geographically labeled to allow the creation of the fingerprint maps.

6.5 Management view

In this section we introduce the *management view*, which specifies the interactions among the entities involved in system administration and maintenance tasks. The main goal behind the definition of this view is to clearly separate the logical specification and implementation of those tasks carried out by the operators as part of the system deployment and maintenance from those purely focused on the sensor scanning and location estimation.

In this view, both the *context model* and the *space model* play an important role, since they are the main sources of information that allow us to carry out the processes hereinafter described. The *application* layer is another key point, this time representing those applications that have been developed in order to accomplish the management tasks. In this case, the *application* layer makes use of the API provided by the *management* layer to access to the *location service* and *sensor manager* functionality as well as to the data provided by the *context model* and the *space model*.

The functionality included within this view has been classified in two different categories:

- **Training methods**: It refers to the functionality of collecting contextual information throughout the scenario using the available sensors (WiFi, images, accelerometer, etc.).
- **Models generation processes**: This category includes those tasks which have been defined with the main aim of supporting the generation of the models used by the different proposals described in this thesis. Among other functionality, it includes the process to build the distribution models based on RSSIs, and the 2D and 3D search structures of images.

In order to show an example of the functionality included in this view, we are going to partially describe the process that takes place during the training phase. During this stage, the operators accomplish different tasks previous to the system deployment, for

example, to obtain fingerprint maps. The logic of these tasks has been included in this view due to the importance of performing the most appropriate design according to the characteristics of each specific environment and the localization requirements imposed. For example, imagine two different systems in which we are going to make use of WiFi and images to estimate the user position, each one of them requiring coarse and fine-grained accuracy to provide scene recognition and AR services respectively. In these cases, the training processes are carried out following different strategies, nevertheless, the functionality used for sensor scanning is always similar.

During the training phase an operator collects contextual information throughout the scenario of deployment making use of the functionality described in the *sensor view*. In this case, the observations collected are geo-tagged in order to build the fingerprint maps. This process can be performed manually by the operator relying on the information provided by the *space model*, or can be performed in an automatic way, making use of solutions like the one we presented in Chapter 5, where we used inertial sensors to infer the path covered by the operator and to perform an automatic labeling of the obtained observations.

Figure 6.4 shows the sequence diagram that represents an example of the flow of information and the communications among the entities which are involved to complete the training process explained in Chapter 5. The process starts when the *application* requires the *management* layer to start the training (step 1 in Figure 6.4). Then the *management* layer starts the sensor scan in the way it was described in previous section and starts the particle filter (step 2). During the sensor data collection, as soon as a new step is detected, the particles evolve according to the estimated location (step 3). Once the operator has completed the walk, the *application* layer notifies (step 4) the *management* layer to stop the sensor scan (step 5). Next, the *management* layer initiates the backward belief propagation process of the still alive particles (step 6) to infer the path covered by the operator. Using the information provided by the particle filter, the *location service* is able to trace the path back to the corresponding starting state (step 7), allowing us to geo-tag each observation with the position where it was collected (step 8). Finally, the set of geo-tagged observations is stored into the repository (step 9).

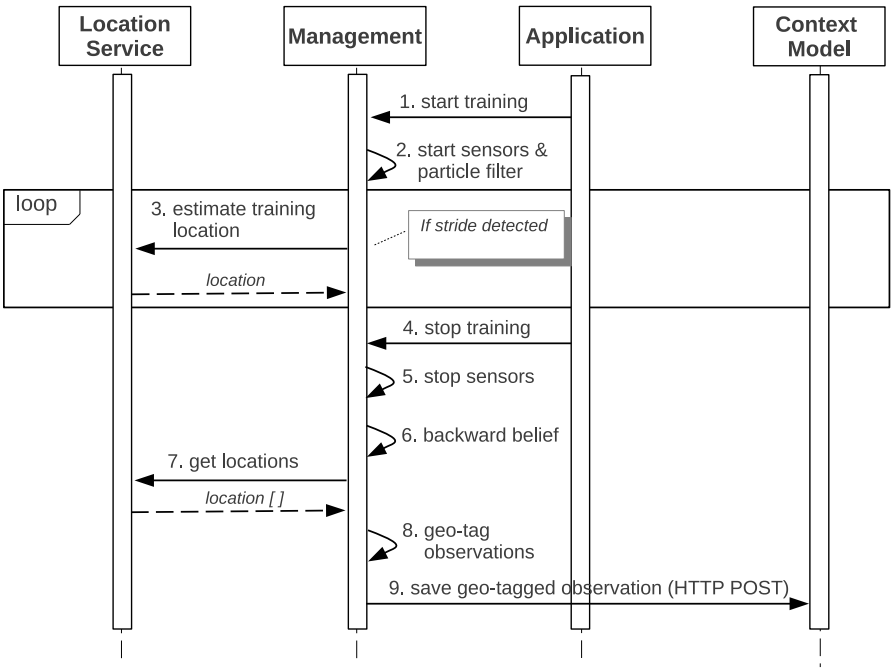


FIGURE 6.4: Sequence diagram representing the process that takes place during training phase.

6.6 Location view

The *location view* defines the functionality to answer the location queries made by external applications. The *query manager* is the main entity of this view since it is in charge of analyzing the requirements indicated by the queries, forwarding them to the most appropriate *location service*. Once the request has been solved, the *query manager* composes the response message from the information provided by the *adaptation* layer, and if required, using the functionality provided by the context model and the space model. The HTTP RESTful API defined by the *query manager* allows the external applications to make use of the services provided.

Our current implementation supports the following three types of queries:

Localization queries: Which estimate the device position within a building complex using the information collected by its sensors. As in the rest of queries, we provide different granularity levels, depending on the accuracy required by the applications and the sensor information available. These queries are the basis of our proposal, since apart of providing position information they also support the resolution of the other types of queries. Localization queries are mainly performed by external location-based applications. Moreover, management applications also make use of this resource, as for

example to estimate the position of the operator during the training phase to geo-tag the measurements.

Navigation queries: These queries solve those situations in which we require the path that should be covered to reach a specific destination, e.g. "the path to reach the library". The response to these queries is formed by a sorted set of locations which the device's owner should follow to reach the desired destination. Physical distance between adjacent points defined by the *space model*, or restrictions related with the navigation flow (e.g. one way doors), are considered in order to provide a useful navigation service. Taking into consideration these factors, we are able to provide two different types of navigations queries, allowing the estimation of the shortest or least-cost path to get to the destination.

Range queries: This type of queries can be considered as a complement to the localization queries, since their main goal is to identify all the entities observed within a specified area considering certain temporal, spatial or circumstantial criteria. To identify all the entities which match the established criteria, e.g., "which of these devices are closer than 50 meters from me?", these queries are addressed considering the information provided by the *space model* and the positioning information of the involved devices, in this case, the requester device and those that might be associated with their friends.

Figure 6.5 shows a sequence diagram that represents the flow of information among layers that take place to solve a query to obtain the last available position of a mobile device. Navigation and range queries are processed in a similar way, thus we omit the inclusion of additional diagrams.

The process starts when an external application makes a query to obtain the location of a specific device using the RESTful HTTP API provided by the *query manager* (step 1). Afterwards, it transforms the HTTP query into a simple call to one of the methods provided by the *adaptation* layer's API (step 2). Then, the *adaptation* layer analyses the requirements imposed by the query and checks whether this query was previously solved and cached estimations can be returned. In this example we consider that no historical information is available. Therefore, the *adaptation* layer obtains the most recent observations associated to the requester device from the *context model* repository (step 3), and the information of the *space model* in which the observations have been obtained (step 4), in case it was not previously loaded. Next, the request is redirected

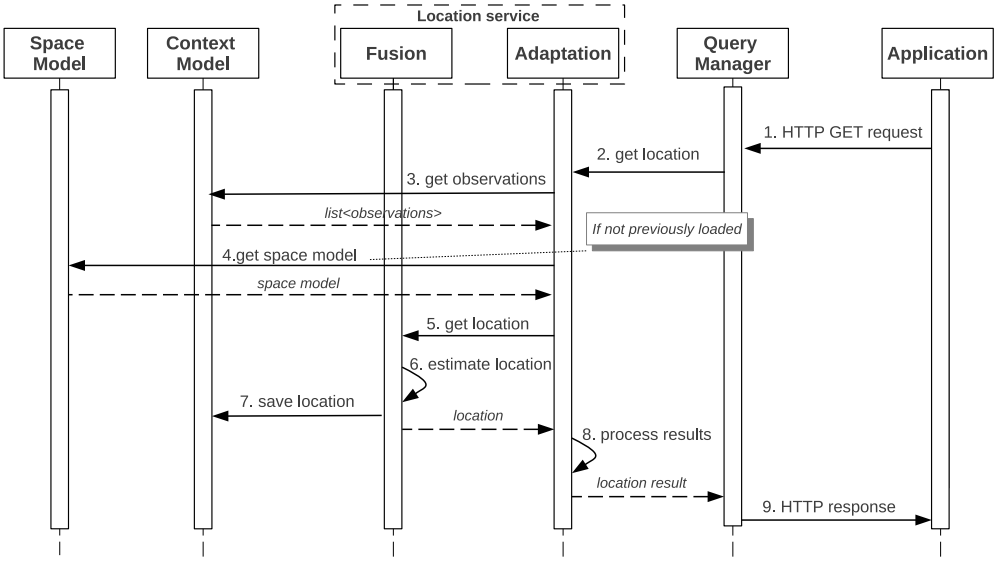


FIGURE 6.5: Localization query processing example.

to the most appropriate instance of the *fusion* layer (step 5), which is in charge of estimating the location of the device (step 6). The result is stored in the repository handled by the *context model*, making it available for future queries (step 7). After processing the result to adjust it to the required accuracy (step 8), it is propagated to the *query manager* (step 9), which builds the HTTP response that is returned to the *application* layer of the requester device.

The following example represents a real case which solves a location query to obtain the last available position of the device identified by the MAC address *12:34:56:78:90:ab*, requiring the finest accuracy. According to our design, in this example we assume that the sensor data are already available at the context model. Next, we indicate the HTTP request made, and the HTTP response together with the result formatted in JSON format:

```
HTTP Request:  GET /query/location?deviceid=12:34:56:78:90:ab&accuracy=precise
HTTP Response: 200 OK
   Content : JSON Schema
  {"deviceId":"12:34:56:78:90:ab",
    "locations":["spModelId": 1,
                 "coords": ["x": 829, "y":-581, "z":164],
                 "rotation":[0.640393,0.175627,-0.747698,
                             -0.767712,0.117584,-0.629915,
                             -0.0227132,0.977409,0.210131],
```

```
                "timestamp": "2014-09-16 12:33:45",
                "accuracy": "precise"]
}
```

- *deviceID*: identifier of the device which has been located.
- *locations*: array containing all the locations which have been obtained, each one them is formed by:
  - *spModelId*: identifies the space model in which the device is located.
  - *coords*: the position of the device, expressed in centimeters from the origin of coordinates.
  - *rotation*: 3×3 matrix which represents the angle of rotation over the tree axis (X, Y, Z).
  - *timestamp*: the moment in which the location was estimated.
  - *accuracy*: the level of accuracy obtained for this request.

6.7 Summary

Throughout this chapter we have discussed the architecture that supports the development of the localization system which has been proposed in the thesis, reaching the goals of extensibility, adaptability, modularity and scalability established at the beginning of this chapter. Though some ideas were indicated in previous chapters, now we have carried out a more detailed description of the distribution of the functionality included. The final design of this architecture is the result of the evolution of our initial proposal. During the last four years, as we observed the new requirements imposed by the integration of additional sensors, we were aware of the need of defining a structural design in favor of the multimodality.

From an abstract point of view, we have introduced the logical entities which were defined to handle the different processes that take place during the life cycle of a location service, that is, from its design (including training and deployment) to its working stage (on-line phase). The proposed architecture allows a clear decoupled implementation of each one of the defined modules, and thus favoring the most appropriate distribution of the functionality according to the system requirements and resources available.

As we indicated at the beginning of this document, the main goal we set when we addressed the design of our architecture was to provide a solution integrating the information obtained from several sensors. This multisensor integration had to enable the appropriate mechanisms to improve the accuracy obtained by existing approaches, being able to adapt to the specific

characteristics of each scenario of deployment, the characteristics of the mobile devices used and the localization requirements imposed. Though it has not been integrated in further systems, a priori we consider that the defined architecture allows to address the issues observed. The ability to adapt to different situations has been demonstrated throughout previous chapters with the different proposals that were made and the experimental tests that were carried out.

Despite being based on existing proposals, our solution provides a different perspective of the way in which the functionality required by a location service is organized. We have proposed a well structured and modular design, enabling the distribution of responsibilities and the workload between the smartphone and the core server. The fact of organizing this functionality in three different views facilitates the integration of new functionality and the inclusion of additional sensors, what goes in favor of the extensibility of the system and opens new opportunities to provide better solutions.

Chapter 7

Conclusions and future work

7.1 Conclusions

In this final chapter we present the main conclusions derived from this work. Throughout this dissertation, but specifically in **Chapter 2**, we did an intensive revision of a wide range of techniques which were proposed to solve different problems regarding the location of mobile devices in indoor environments. We analyzed the pros and cons of each, identifying the possibilities of integration to develop a better solution. This enabled us to present different proposals which define a multisensor location service taking advantage of the information collected by different sensors integrated in common smartphones.

One of our initial goals was to define a solution able to provide location information at different levels of accuracy. In **Chapter 3**, we proposed the utilization of several existing techniques that made use of WiFi signals to provide location information. Combining them in an appropriate way, we were able to provide coarse-grained positions (room-size accuracies) of the smartphone with a relatively good reliability in our estimations. However we realized that to obtain more accurate estimations and to improve the reliability, we had to rely on the integration of additional sensors.

Therefore, we introduced the use of images within our positioning algorithm, described in **Chapter 4**. This process was accomplished in two different steps:

- Initially we made use of the images to carry out a process of *scene recognition*. This technique was based on the utilization of fingerprint maps of images, which ensured a better reliability of the estimations obtained in comparison with the cases in which only RSSIs were used. However it still did not facilitate the estimation of very precise 3D locations.

- During the next step we introduced the use of *3D maps* of the environment to support the application of geometric techniques in computer vision. This allowed us to estimate the exact 3D position and orientation of the smartphone.

The use of WiFi and images allowed the possibility of providing different levels of accuracy, each having its own advantages and disadvantages. When using WiFi, the main issue was its deficiency to obtain high accurate estimations. In the case of using images, though improving the system accuracy, the main drawback was the time required to perform the image analysis, which did not enable us to obtain position information in real time (according to the camera preview frame rate at smartphones). In order to minimize the processing time of images, we proposed an initial multisensor integration which accomplished the localization process in two different phases:

- In the first phase we analyzed the RSSIs. The low latency of calculating the location analyzing these radio signals, allowed us to obtain an initial coarse-estimation in just a few milliseconds. Sometimes, the accuracy obtained can be considered good enough in those situations in which there is no information available from images, or simply when the application does not require a better accuracy.
- In a second step, we constrained the search space to accomplish the image matching process. In this case, the image analysis was focused on a limited area of the scenario based on the position estimation obtained using the RSSIs, thus reducing the time required to complete this task.

According to the different tests that we carried out, our proposal ensured a good trade-off between the accuracy obtained (less than 5 cm of estimation error) and the response time (around 250 ms). However, we are aware that the response time achieved is still far from the goal set at the beginning of our work, namely to support real time applications. We tried to minimize this shortcoming by integrating the gyroscope, which was used to correct the position of the device when it performed smooth movements, until a new estimation was obtained.

Another important contribution was the multisensor solution which addressed a more efficient and mostly unsupervised way to accomplish the training phase of fingerprint based systems, described in **Chapter 5**. The analysis of inertial sensors measurements obtained while performing the training enabled the possibility of tracing the route followed by the operators. The inferred path allowed the automatic geo-tagging of the measurements obtained by other sensors (WiFi and camera), and facilitated the unsupervised creation of fingerprint maps of radio signals and images. As we demonstrated, this process resulted in more reliable reconstructions of the 3D maps of the scenes.

The use of geographically labeled images avoided common mistakes in 3D maps creation which affect the process of reconstruction, for example those caused by finding coincidences between similar objects present at the scene but physically distant. As we discussed, the main challenge of this proposal was to make use of the inertial sensors integrated in smartphones. In our case we had to deal with the specific restrictions regarding the device pose and the movements patterns in order to ensure the collection of clear pictures.

We consider our proposal an initial step in the resolution of this complex work. Though we have obtained good results, we are aware of the different restrictions we had to make in order to accomplish this task. For example, the motion pattern that operators have to follow in order to detect their strides and movement orientation, is a clear drawback that must be addressed in the future. Moreover, we are also conscious that our test bed environments clearly favored the application of this solution. In the future, it would be necessary to evaluate its performance in environments with more complicated designs.

Finally, another worthy contribution has been the proposal of a distributed architecture presented in **Chapter 6**, which facilitates the integration of the different components that take part in the localization process. We find our proposal very interesting for defining a localization system, especially regarding the distribution of functionality between the different elements that take part in it. Though we reckon that our proposal covers a vast number of possible configurations, including the management of different sensor measurements and contextual information, it still might require additional modifications to adapt to specific needs. In any case, we consider it a good starting point for developers in order to help them to obtain a clear perspective of the organizational requirements implied in the design of a location service.

7.1.1 Scientific publications

In this section we summarize the scientific publications which have been produced during our research. They have been published or are under revision on several international journals and international conferences. We also include the publications made as collaboration with the Department of Computer Science of the University of Aarhus (Denmark).

International Journals

1. A.J. Ruiz-Ruiz, O. Canovas, and P.E. Lopez-de-Teruel. *A multisensor architecture providing location-based services for smartphones.* In Journal of Mobile Networks and Applications, *18(3):310-325*, 2012.

2. A.J. Ruiz-Ruiz, O. Canovas, and P.E. Lopez-de-Teruel. *A vision-enhanced multisensor LBS suitable for augmented reality applications.* In Journal of Location Based Services, *7(3):145-164*, 2013.
3. T.S. Prentow, A.J. Ruiz-Ruiz, H. Blunck, A. Stisen and M.B. Kjærgaard. *Spatio-temporal Facility Utilization Analysis from Exhaustive WiFi Monitoring.* In Journal of Pervasive and Mobile Computing, 2014 (Submitted).

Conference proceedings

1. A.J. Ruiz-Ruiz, O. Canovas. *Integrating probabilistic techniques for indoor localization of heterogeneous clients.* En Jornadas de Ingeniería Telemática (JITEL), (Santander, Spain , 2011).
2. A.J. Ruiz-Ruiz, O. Canovas, R.A. Rubio, and P.E. Lopez-de-Teruel. *Using SIFT and WiFi signals to provide location-based services for smartphones.* In Proceedings of the International Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS), (Copenhaguen, Denmark, 2011), pp. 37-48.
3. A.J. Ruiz-Ruiz, P.E. Lopez-de-Teruel and O. Canovas. *A multisensor LBS using SIFT-based 3D models.* In Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN), (Sydney, Australia, 2012), pp. 1-10.
4. A.J. Ruiz-Ruiz, O. Canovas, and P.E. Lopez-de-Teruel. *Practical image-enhanced LBS for AR applications.* In Proceedings of the International Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS), (Tokyo, Japan, 2013), pp. 460-473.
5. A.J. Ruiz-Ruiz, H. Blunck, T.S. Prentow, A. Stisen and M.B. Kjærgaard. *Analysis methods for extracting knowledge from large-scale WiFi monitoring to inform building facility planning.* In Proceedings of the International Conference on Pervasive Computing and Communications (PerCom), (Budapest, Hungary, 2014), pp. 130-138.

7.2 Future work

After the work carried out during our research, we identify different points that should be addressed in the future. Some of them are related to the improvement of our proposal, in order to solve the different shortcomings observed. Additionally, other lines of work have been discovered throughout the development of this thesis, and we consider them equally interesting from the perspective of its integration within our current solution.

Regarding the work already addressed in this thesis, we find the following areas in need of improvement:

- Firstly, our main goal for future research is to reduce the time required to carry out a location estimation combining WiFi and images. As we indicated, our current solution is not able to support those applications requiring high updating rates. Meanwhile, we try to solve this drawback correcting the device position according to the information provided by the gyroscope, but we are aware of the limitations of this solution.
- A still pending goal is the integration of inertial sensors in the on-line phase. Currently, this integration is limited to the detection of static and motion stages, what optimizes the transmissions between the smartphone and the core server. However, our idea is to support the localization process during short periods of time in which the smartphone would not constantly request the core server to calculate its position.
- It could also be interesting to integrate the image features extraction and matching processes into the smartphone. The recent development of these devices, which have increased the computational capabilities, makes them feasible to carry out these tasks. This may need alternative techniques for the extraction and matching of features, less intensive than the ones used in our current solution. Several members of our research group are already working on the integration of lightweight features that could provide similar results to the technique currently used.
- Another interesting research line to work on is to compress the generated 3D models, obtaining more reduced representation of the environments, and therefore minimizing the computational demands to perform the matching process. The main goal would be to supply the smartphones with the 3D models that represent their current environment, making them able to carry out this work locally.
- Finally, it could be developed some alternative to the method to perform the automatic geo-tag of measurements during the training phase. Our current method is quite restrictive regarding the motion pattern to be followed by the operator, justified by the poor quality of inertial sensor integrated in the smartphones and the requirements regarding the device pose to collect clear images. However we found the possibility to evolve this solution very interesting, enabling the collection of images using video recording and providing an alternative to estimate the relative position of the operator during the training period.

Another set of future research lines consists on the analysis and integration of additional functionality that may help to build a more complete location service:

- We have proposed a feasible way to perform an almost automatic and unsupervised training, but maintenance of the system models is still an important challenge. From that point of view, it would be interesting to design an optimal solution that supports the continuous updating of current models, both WiFi and image based 3D. This solution should be

based on already existing approaches which use crowdsensing techniques, but taking into account the additional constraints related to the management of images.

- Another important weakness in our proposal is the lack in the integration of security and privacy techniques. For that reason, one of our main future goals is to provide the appropriate methods to ensure the security of the data managed by our system. On the one hand, we have to minimize the data transmission between the mobile phone and the core server, thus reducing the risk of being monitored. The data transmitted and stored in the repository should be well encrypted. On the other hand, to maintain the privacy of the users, it would be necessary to integrate some mechanisms that enable their anonymity. Although several proposals are available, like the one made by Huertas et al. [91], we should perform an in-depth analysis in order to assess their usefulness within our architecture, introducing the corresponding modifications to provide an optimal solution.

As we can observe, there is clearly much work to be done. The wide range of applicability of this type of systems and the continuous evolution of smartphones forces the constant development of new alternatives integrating new characteristics and adapting its functionality to the emerging necessities. Some very exciting developments are underway in this field of research, which will no doubt experiment amazing advances in the near future.

Bibliography

[1] Aeroscout company. http://www.aeroscout.com. Accessed: 2014-10-13.

[2] iBeacon. https://developer.apple.com/ibeacon. Accessed: 2014-10-13.

[3] Ingress. https://www.ingress.com. Accessed: 2014-10-13.

[4] Insoft. http://www.infsoft.com. Accessed: 2014-10-13.

[5] Meridian, aruba networks. http://www.meridianapps.com. Accessed: 2014-10-13.

[6] MvixAir: Bluetooth Marketing System. http://goo.gl/LO8zd8. Accessed: 2014-10-13.

[7] Real-time location system. http://www.ekahau.com. Accessed: 2014-10-13.

[8] Sensewhere. http://www.sensewhere.com. Accessed: 2014-10-13.

[9] SiftGPU: A GPU implementation of scale invaraint feature transform (SIFT). http://cs.unc.edu/~ccwu/siftgpu. Accessed: 2014-10-13.

[10] Tango project, Google. https://www.google.com/atap/projecttango. Accessed: 2014-10-13.

[11] VisualSFM: A visual structure from motion system. http://www.cs.washington.edu/homes/ccwu/vsfm. Accessed: 2014-10-13.

[12] Indoor location market: Global advancements, market forecasts and analysis (2013 - 2018). Technical report, Markets and Markets, 2013.

[13] Our mobile planet, Google. Technical report, Ipsos Media CT, 2013.

[14] Indoor location smartphone applications. Technical report, ABI Research, 2014.

[15] U. Ahmad, A. Gavrilov, U. Nasir, M. Iqbal, S. Jin, and C. S. Lee. In-building localization using neural networks. In *Proceeding of Engineering of Intelligent Systems*, pages 1–6, 2006.

[16] V. Amendolare, D. Cyganski, R. Duckworth, S. Makarov, J. Coyne, H. Daempfling, and B. Woodacre. WPI precision personnel locator system: Inertial navigation supplementation. In *Proceedings of the Position, Location and Navigation Symposium*, pages 350–357, 2008.

[17] S. Aparicio, J. Pérez, P. Tarrío, A. Bernardos, and J. Casar. An indoor location method based on a fusion map using Bluetooth and WLAN technologies. In *International Symposium on Distributed Computing and Artificial Intelligence (DCAI)*, pages 702–710. 2009.

[18] I. Arai, S. Horimi, and N. Nishio. Wi-Foto 2: Heterogeneous device controller using Wi-Fi positioning and template matching. In *Proceedings of the International Conference on Pervasive Computing (Pervasive)*, 2010.

[19] W. Arbaugh, M. Shin, and A. Mishra. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *ACM SIGCOMM Computer Communications Review*, 33(2):93–102, 2003.

[20] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[21] D. Aufderheide and W. Krybus. Towards real-time camera egomotion estimation and three-dimensional scene acquisition from monocular image streams. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, 2010.

[22] M. Azizyan, I. Constandache, and R. R. Choudhury. Surroundsense: Mobile phone localization via ambience fingerprinting. In *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 69–72, 2009.

[23] P. Bahl and V. N. Padmanabhan. Enhancements to the RADAR user location and tracking system. Technical report, Microsoft Corp., Tech. Rep. MSR-TR-2000, 2000.

[24] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the International Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 775–784, 2000.

[25] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner. Virtual compass: relative positioning to sense mobile social interactions. In *Proceedings of the International Conference on Pervasive Computing (PerCom)*, pages 1–21, 2010.

[26] O. Barnich and M. Van Droogenbroeck. Vibe: A powerful random technique to estimate the background in video sequences. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 945–948, 2009.

[27] R. Battiti, N. T. Le, and A. Villani. Location-aware computing: a neural network model for determining location in wireless LANs. Technical report, University of Trento, 2002.

[28] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[29] S. Beauregard and H. Haas. Pedestrian dead reckoning: A basis for personal positioning. In *Proceedings of the Workshop on Positioning, Navigation and Communication (WPNC)*, pages 27–35, 2006.

[30] C. Becker and F. Durr. On location models for ubiquitous computing. *Personal Ubiquitous Computing*, 9(1):20–31, 2005.

[31] J. Beis and D. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1000–1006, 1997.

[32] A. Bekkelien and M. Deriaz. Harmonization of position providers. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–4, 2012.

[33] S. Bhattacharya. Enriching location information: An energy-efficient approach. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp)*, pages 519–522, 2011.

[34] G. Blázquez Gil, A. Berlanga, and J. M. Molina. InContexto: multisensor architecture to obtain people context from smartphones. *International Journal of Distributed Sensor Networks*, 2012:12, 2012.

[35] G. Borriello, M. Chalmers, A. LaMarca, and P. Nixon. Delivering real-world ubiquitous location systems. In *Communications of the ACM*, pages 36–41, 2005.

[36] A. Bose and C. H. Foh. A practical path loss model for indoor WiFi positioning enhancement. In *Proceedings of the International Conference on Information, Communications and Signal Processing (ICICS)*, pages 1–5, 2007.

[37] P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *Proceedings of the International Conference on Computer Vision (ECCV)*, pages 350–362. 2004.

[38] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[39] L. Castro and J. Favela. Continuous tracking of user location in WLANs using recurrent neural networks. In *Proceedings of the International Conference on Computer Science (ENC)*, pages 174–181, 2005.

[40] F. Cavallo, A. Sabatini, and V. Genovese. A step toward GPS/INS personal navigation systems: real-time assessment of gait by foot inertial sensing. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1187–1191, 2005.

[41] L. Chen, E. Wu, J. Ming, and G. Chen. Homogeneous features utilization to address the device heterogeneity problem in fingerprint localization. *IEEE Sensors Journal*, 14(4):998–1005, 2014.

[42] H. Cheng, H. Luo, and F. Zhao. Device-clustering algorithm in crowdsourcing-based localization. *The Journal of China Universities of Posts and Telecommunications*, 19(2):114 – 121, 2012.

[43] P. Cheong, A. Rabbachin, J. Montillet, K. Yu, and I. Oppermann. Synchronization, TOA and position estimation for low-complexity LDR UWB devices. In *Proceedings of the International Conference on Ultra-Wideband (ICUWB)*, pages 480–484, 2005.

[44] S. Chess. Augmented regionalism: Ingress as geomediated gaming narrative. *Information, Communication and Society*, 17(9):1105–1117, 2014.

[45] J. Chey, J. W. Chun, J. G. Lee, and G. I. Jee. Indoor positioning using wireless LAN signal propagation model and reference points. In *Proceedings of the National Technical Meeting of The Institute of Navigation*, pages 1107–1112, 2001.

[46] B. S. Cho, W. Moon, W. J. Seo, and K. R. Baek. A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. *Journal of Mechanical Science and Technology*, 25(11):2907–2917, 2011.

[47] J. Chon and H. Cha. Lifemap: a smartphone-based context provider for location-based services. *IEEE Pervasive Computing*, 10(2):58–67, 2011.

[48] Y. Chunhua, H. Yi, and Z. Xu. Hybrid TDOA/AOA method for indoor positioning systems. In *IET Seminar on Location Technologies*, pages 1–5, 2007.

[49] D. Colomar, J. Nilsson, and P. Handel. Smoothing for ZUPT-aided INSs. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–5, 2012.

[50] P. Davidson, J. Collin, and J. Takala. Application of particle filters for indoor positioning using floor plans. In *Proceedings of the International Conference on Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, pages 1–4, 2010.

[51] N. Dragos and B. Nath. Ad-hoc positioning system (APS) using AOA. In *Proceedings of the IEEE Computer and Communications Societies (InfoCom)*, pages 1734–1743, 2003.

[52] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox. Interactive 3D modeling of indoor environments with a consumer depth camera. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp)*, pages 75–84, 2011.

[53] M. Enzweiler and D. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009.

[54] Eurostat. *How europeans spend their time. Everyday life of women and men.* European Communties, 2004.

[55] F. Evennou and F. Marx. Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning. *Eurasip journal on applied signal processing*, 2006:164–164, 2006.

[56] D. Fahed and R. Liu. Wi-Fi-based localization in dynamic indoor environment using a dynamic neural network. *International Journal of Machine Learning and Computing*, 3(1):127–131, 2013.

[57] R. Faragher, C. Sarno, and M. Newman. Opportunistic radio SLAM for indoor navigation using smartphone sensors. In *Position Location and Navigation Symposium (PLANS)*, pages 120–128, 2012.

[58] C. Fernandes, M. Campos, and L. Chaimowicz. A low-cost localization system based on artificial landmarks. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS)*, pages 109–114, 2012.

[59] F. D. Ferris, B. and N. D. Lawrence. WiFi-SLAM using gaussian process latent variable models. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 2480–2485, 2007.

[60] P. Fiore. Efficient linear solution of exterior orientation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):140–148, 2001.

[61] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[62] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[63] F. Frauendorfer, C. Wu, J. M. Frahm, and M. Pollefeys. Visual word based location recognition in 3D models using distance augmented weighting. In *Proceedings of the International Conference on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008.

[64] A. Furlan, S. Miller, D. G. Sorrenti, L. Fei-fei, and S. Savarese. Free your camera: 3D indoor scene understanding from arbitrary camera motion. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12, 2013.

[65] V. Garcia, E. Debreuve, and M. Barlaud. Fast k-nearest neighbor search using GPU. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, 2008.

[66] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 620–629, 2005.

[67] H. Gene, C. Golub, and F. V. Loan. *Matrix Computations*. Johns Hopkins Studies in Mathematical Sciences, 3 edition, 1996.

[68] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2 edition, 2002.

[69] N. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Proceedings F on Radar and Signal Processing*, pages 107–113, 1993.

[70] J. C. Gower and G. B. Dijksterhuis. *Procrustes Problems*. Oxford University Press, 2004.

[71] D. Graumann, W. Lara, J. Hightower, and G. Borriello. Real-world implementation of the location stack: the universal location framework. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 122–128, 2003.

[72] W. Griswold, P. Shanahan, S. Brown, R. Boyer, M. Ratto, R. Shapiro, and T. Truong. ActiveCampus: experiments in community-oriented ubiquitous computing. *IEEE Computer*, 37(10):73–81, 2004.

[73] I. Guvenc and C. C. Chong. A survey on TOA based wireless localization and NLOS mitigation techniques. *IEEE Communications Surveys and Tutorials*, 11(3):107–124, 2009.

[74] Y. Gwon and R. Jain. Error characteristics and calibration-free techniques for wireless LAN-based location estimation. In *Proceedings International Workshop on Mobility Management and Wireless Access Protocols (MobiWac)*, pages 2–9, 2004.

[75] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the International Conference on Mobile computing and networking (MobiCom)*, pages 70–84, 2004.

[76] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2):187–197, 2002.

[77] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2011.

[78] H. Hashemi. The indoor radio propagation channel. *Proceedings of the IEEE*, 81(7):943–968, 1993.

[79] K. Hattori, R. Kimura, N. Nakajima, T. Fujii, Y. Kado, B. Zhang, T. Hazugawa, and K. Takadama. Hybrid indoor location estimation system using image processing and WiFi strength. In *Proceedings of the International Conference on Wireless Networks and Information Systems (WNIS)*, pages 406–411, 2009.

[80] S. Haykin. *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[81] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[82] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-d mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Experimental Robotics*, pages 477–491. 2014.

[83] J. Hightower, B. Brumitt, and G. Borriello. The location stack: a layered model for location in ubiquitous computing. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 22–28, 2002.

[84] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System. Theory and Practice.* Springer, 2000.

[85] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer, 2007.

[86] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Proceedings of the International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, pages 194–205, 2005.

[87] J. I. Hong and J. A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 177–189, 2004.

[88] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[89] H. Hu and D. L. Lee. Semantic location modeling for location navigation in mobile environment. In *Proceedings of the International Conference on Mobile Data Management (MDM)*, pages 52–61, 2004.

[90] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal. Efficient, generalized indoor WiFi graphslam. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1038–1043, 2011.

[91] A. Huertas Celdran, F. Garcia Clemente, M. Gil Perez, and G. Martinez Perez. Secoman: A semantic-aware policy framework for developing privacy-preserving and context-aware smart applications. *IEE Systems Journal*, PP(99):1–14, 2014.

[92] T. Hui, W. Shuang, and X. Huaiyao. Localization using cooperative AOA approach. In *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 2416–2419, 2007.

[93] J. Ido, Y. Shimizu, Y. Matsumoto, and T. Ogasawara. Indoor navigation for a humanoid robot using a view sequence. *The International Journal of Robotics Research*, 28:315–325, 2009.

[94] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *Proceedings of the Workshop on Motion and Video Computing (MOTION)*, pages 22–28, 2002.

[95] K. Jeong and H. Moon. Object detection using FAST corner detector based on smartphone platforms. In *Proceedings of the International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI)*, pages 111–115, 2011.

[96] C. Jiang and P. Steenkiste. A hybrid location model with a computable location identifier for ubiquitous computing. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp)*, pages 246–263, 2002.

[97] Y. Jiang, X. Pan, K. Li, Q. Lv, R. P. Dick, M. Hannigan, L. Shang, and A. Arbor. ARIEL: Automatic Wi-Fi based room fingerprinting for indoor localization. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp)*, pages 441–450, 2012.

[98] A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *Advances in Spatial and Temporal Databases*, pages 239–257. 2007.

[99] T. King, Haenselmann, and W. Effelsberg. Deployment, calibration, and measurement factors for position errors in 802.11-based indoor positioning systems. In *Proceedings of the International Conference on Location-and Context-Awareness (LoCA)*, pages 17–34, 2007.

[100] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg. COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses. In *Proceedings of the International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, pages 34–40, 2006.

[101] M. Kjærgaard. Automatic mitigation of sensor variations for signal strength based location systems. In *Proceedings of the International Symposium on Location and Context Awareness (LoCA)*, pages 30–47, 2006.

[102] M. B. Kjærgaard and C. V. Munk. Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength. In *Proceedings of the International Conference on Pervasive Computing and Communications (Percom)*, pages 110–116, 2008.

[103] M. B. Kjærgaard, G. Treu, P. Ruppel, and A. Küpper. Efficient indoor proximity and separation detection for location fingerprinting. In *Proceedings of the International Conference on Mobile Wireless Middleware, Operating Systems, and Applications (MobilWare)*, 2008.

[104] M. B. Kjærgaard, M. Wirz, D. Roggen, and G. Tröster. Detecting pedestrian flocks by fusion of multi-modal sensors in mobile phones. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp)*, pages 240–249, 2012.

[105] K. Konolige and K. Chou. Markov localization using correlation. pages 1154–1159, 1999.

[106] V. Kostakos, T. Kindberg, A. F. G. Schiek, A. Penn, D. S. Fraser, and T. Jones. Instrumenting the city: Developing methods for observing and understanding the digital cityscape. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp)*, pages 315–332, 2006.

[107] M. Kourogi and T. Kurata. Personal positioning based on walking locomotion analysis with self-contained sensors and a wearable camera. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, page 103, 2003.

[108] B. Krach and P. Robertson. Integration of foot-mounted inertial sensors into a bayesian location estimation framework. In *Proceeding of the International Workshop on Positioning, Navigation and Communication (WPNC)*, pages 55–61, 2008.

[109] P. Krishnan, A. Krishnakumar, W. Ju, C. Mallows, and S. Ganu. A system for LEASE: Location estimation assisted by stationary emitters for indoor RF wireless networks. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[110] S. Kristoffer, D. Galvez Lopez, P. Chandana, P. Jensfelt, and D. Kragic. Object search and localization for an indoor mobile robot. *Journal of Computing and Information Technology - CIT*, 17(1):67–80, 2009.

[111] J. Krumm. The nearme wireless proximity server. pages 283–300, 2004.

[112] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for easyliving. In *Proceedings of the International Workshop on the Visual Surveillance*, pages 3–10, 2000.

[113] J. Krumm and E. Horvitz. Locadio: inferring motion and location fron Wi-Fi signal strengths. pages 4–13, 2004.

[114] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, D. S. Wallach, and E. Kavraki. Using wireless ethernet for localization. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 402–408, 2002.

[115] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, D. S. Wallach, and G. Marceau. Robotics-based location sensing using wireless ethernet. In *Proceedings of the International Conference on Mobile computing and Networking (MobiCom)*, pages 227–238, 2002.

[116] A. M. Ladd, K. E. Bekris, A. P. Rudys, D. S. Wallach, and L. E. Kavraki. On the feasibility of using wireless ethernet for indoor localization. *IEEE Transactions on Robotics and Automation*, 20(3):555–559, 2004.

[117] A. LaMarca, Y. Chawathe, S. Consolvo, et al., and J. Hightower. Place lab: Device positioning using radio beacons in the wild. In *Proceedings of the International Conference on Pervasive Computing and Communications (Percom)*, pages 116–133, 2005.

[118] C. Laoudias, D. Zeinalipour-Yazti, and C. Panayiotou. Crowdsourced indoor localization for diverse devices through radiomap fusion. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, 2013.

[119] H. Lemelson, M. B. Kjærgaard, R. Hansen, and T. King. Error estimation for indoor 802.11 location fingerprinting. In *Proceedings of the International Symposium on Location and Context Awareness (LoCA)*, pages 138–155, 2009.

[120] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.

[121] J. Li and N. M. Allinson. A comprehensive review of current local features for computer vision. *Neurocomputing*, 71(10):1771–1787, 2008.

[122] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian. Foreground object detection from videos containing complex background. In *Proceedings of the International Conference on Multimedia*, pages 2–10, 2003.

[123] X. Li and J. Wang. Image matching techniques for vision-based indoor navigation systems: performance analysis for 3D map based approach. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, 2012.

[124] X. Li and J. Wang. Multi-image matching for 3D mapping in vision-based navigation applications. In *Proceedings of the International Conference on Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS)*, pages 1–8, 2012.

[125] T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, and A. Zakhor. Indoor localization and visualization using a human-operated backpack system. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, 2010.

[126] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[127] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[128] M. Martinez, F. Villanueva, M. Santofímia, and J. Lopez. A multimodal distributed architecture for indoor localization. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–4, 2011.

[129] L. Medsker and L. C. Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.

[130] J. Meyerowitz and R. R. Choudhury. Hiding stars with fireworks: Location privacy through camouflage. In *Proceedings of the International Conference on Mobile Computing and Networking (Mobicom)*, pages 345–356, 2009.

[131] L. E. Miller. *Indoor navigation for first responders: a feasibility study*. Wireless Communication Technologies Group, Advanced Networking Technologies Devision, Information Technology Laboratory, National Institute of Standards and Technology, 2006.

[132] P. Misra and P. Enge. *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[133] T. Miyaki, T. Yamasaki, and K. Aizawa. Tracking persons using particle filter fusing visual and Wi-Fi localizations for widely distributed camera. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 225–228, 2007.

[134] A. Moller, M. Kranz, R. Huitl, S. Diewald, and L. Roalter. A mobile indoor navigation system interface adapted to vision-based localization. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia (MUM)*, pages 1–10, 2012.

[135] D. M. Mount and S. Arya. ANN, a library for approximate nearest neighbor searching. In *Proceedings of CGC Workshop on Computational Geometry*, 1997.

[136] A. Muhammad, M. Mazliham, P. Boursier, and M. Shahrulniza. K-nearest neighbor algorithm for improving accuracy in clutter based location estimation of wireless nodes. *Malaysian Journal of Computer Science*, pages 146–159, 2011.

[137] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, pages 331–340, 2009.

[138] A. Mulloni, D. Wagner, I. Barakonyi, and D. Schmalstieg. Indoor positioning and navigation with camera phones. *IEEE Pervasive Computing*, 8(2):22–31, 2009.

[139] A. C. Murillo, J. Guerrero, and C. Sagues. SURF features for efficient robot localization with omnidirectional images. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3901–3907, 2007.

[140] K. Muthukrishnan, M. Lijding, and N. Meratnia. Sensing motion using spectral and spatial analysis of wlan rssi. pages 62–76, 2007.

[141] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[142] J. G. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie. Growing an organic indoor location system. In *Proceedings of the International Conference on Mobile systems, applications, and services (MobiSys)*, pages 271–284, 2010.

[143] S. Phaiboon. An empirically based path loss model for indoor wireless channels in laboratory building. In *Proceedings in the Conference on Computers, Communications, Control and Power Engineering (TENCON)*, pages 1020–1023, 2002.

[144] A. Pratama, Widyawan, and R. Hidayat. Smartphone-based pedestrian dead reckoning as an indoor positioning system. In *Proceedings in the International Conference on System Engineering and Technology (ICSET)*, pages 1–6, 2012.

[145] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the International Conference on Mobile computing and networking (MobiCom)*, pages 32–43, 2000.

[146] L. Radaelli and C. S. Jensen. Towards fully organic indoor positioning. In *Proceedings of the International Workshop on Indoor Spatial Awareness (SIGSPATIAL)*, pages 16–20, 2013.

[147] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the International Conference on Mobile computing and networking (MobiCom)*, pages 293–304, 2012.

[148] I. Ramani and S. Savage. SyncScan: practical fast handoff for 802.11 infrastructure networks. In *Proceedings of the International Conference on Computer and Communications Societies (InfoCom)*, pages 675–684, 2005.

[149] T. S. Rappaport. *Wireless Communications-Principles and Practice*. Prentice Hall Communications Engineering and Emerging Technologies, 2005.

[150] L. Richardson and S. Ruby. *Restful web services*. O'Reilly, 2007.

[151] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1508–1515, 2005.

[152] A. J. Ruiz-Ruiz, H. Blunk, T. S. Prentow, A. Stisen, and M. B. Kjærgaard. Analysis methods for extracting knowledge from large-scale WiFi monitoring to inform building facility planning. In *Proceeding of the International Conference on Pervasive Computing and Communications (Percom)*, pages 130–138, 2014.

[153] S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat. Location determination of a mobile device using ieee 802.11b access point signals. In *Proceeding of the International Conference on Wireless Communications and Networking (WCNC)*, pages 1987–1992, 2003.

[154] M. Saputra, W. Widyawan, G. Putra, and P. Santosa. Indoor human tracking application using multiple depth-cameras. In *Proceedings of the International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 307–312, 2012.

[155] J. Schmid, T. Gadeke, D. Curtis, and J. Ledlie. Improving sparse organic WiFi localization with inertial sensors. In *Proceedings of the International Workshop on Positioning Navigation and Communication (WPNC)*, pages 30–35, 2012.

[156] S. Se, D. Lowe, and J. Little. Vision based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, 2005.

[157] S. Y. Seidel and T. S. Rappaport. 914 MHz path loss prediction models for indoor wireless communications in multifloored buildings. *IEEE Transactions on Antennas and Propagation*, 40(2):207–217, 1992.

[158] I. Sharp and K. Yu. Sensor-based dead-reckoning for indoor positioning. *Physical Communication*, 13:1–10, 2013.

[159] X. Shen, K. Xu, X. Sun, J. Wu, and J. Lin. Optimized indoor wireless propagation model in WiFi-RoF network architecture for rss-based localization in the internet of things. In *Proceedings of the International Topical Meeting on Microwave Photonics*, pages 274–277, 2011.

[160] C.-F. Shu, A. Hampapur, M. Lu, L. Brown, J. Connell, A. Senior, and Y. Tian. IBM smart surveillance system (S3): a open and extensible framework for event based surveillance. In *Proceedings of the Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 318–323, 2005.

[161] A. Smailagic and D. Kogan. Location sensing and privacy in a context-aware computing environment. *IEEE Wireless Communications*, 9(5):10–17, 2002.

[162] A. Smailagic, J. Small, and D. P. Siewiorek. Determining user location for context aware computing through the use of a wireless LAN infrastructure. *Institute for Complex Engineered Systems Carnegie Mellon University*, 15213, 2000.

[163] L. Snidaro, C. Micheloni, and C. Chiavedale. Video security for ambient intelligence. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):133–144, 2005.

[164] J. So, J. Y. Lee, C. H. Yoon, and H. Park. An improved location estimation method for WiFi fingerprintbased indoor localization. *International Journal of Software Engineering Its Applications*, 7(3):77, 2013.

[165] G. Stevenson, J. Ye, S. Dobson, and P. Nixon. LOC8: A location model and extensible framework for programming with location. *IEEE Pervasive Computing*, 9(1):28–37, 2010.

[166] R. Stirling, J. Collin, K. Fyfe, and G. Lachapelle. An innovative shoe-mounted pedestrian navigation system. In *Proceedings of European navigation conference (GNSS)*, pages 110–115, 2003.

[167] K. Sung and H. Kim. Bayesian navigation system with particle filtering and dead reckoning in urban canyon environments. In *Proceedings of the International Conference on Sensor, Mesh and Ad Hoc Communications and Networks(SECON)*, pages 73–75, 2012.

[168] M. J. Swain and D. H. Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[169] h. . `http://goo.gl/uDxvAJ`. n. . A. Tech-Thought Inc], title =Smartphone Market Share By Country.

[170] T. Teixeira, D. Jung, G. Dublon, and A. Savvides. Identifying people in camera networks using wearable accelerometers. In *Proceedings of the International Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, pages 1–8, 2009.

[171] T. Teixeira, D. Jung, and A. Savvides. Tasking networked cctv cameras and mobile phones to identify and localize multiple people. In *Proceedings of the International Conference on Ubiquitous computing (Ubicomp)*, pages 213–222, 2010.

[172] S. Thrun, W. Burgard, D. Fox, et al. *Probabilistic robotics*. MIT press Cambridge, MA, 2005.

[173] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 273–280, 2003.

[174] E. Trevisani and A. Vitaletti. Cell-id location technique, limits and benefits: an experimental study. In *Proceedings of the International Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 51–60, 2004.

[175] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, 2000.

[176] P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Proceedings of the Computer Vision Workshops (ICCV Workshops)*, pages 2109–2116, 2009.

[177] A. Vedaldi and S. Soatto. Features for recognition: viewpoint invariance for non-planar scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1474–1481, 2005.

[178] H. Velayos and G. Karlsson. Techniques to reduce the IEEE 802.11b handoff time. In *Proceedings of the International Conference on Communications (ICC)*, pages 3844–3848, 2004.

[179] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001.

[180] G. Wang and S. Shen. An efficient localization algorithm using sparse anchors for the internet of things. In *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 371–378, 2013.

[181] R. Want and A. Hopper. Active badges and personal interactive computing objects. *IEEE Transactions on Consumer Electronics*, 38(1):10–20, 1992.

[182] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[183] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[184] M. Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.

[185] M. Werner, M. Kessel, and C. Marouane. Indoor positioning using smartphone camera. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6, 2011.

[186] O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp)*, pages 114–123, 2008.

[187] B. Xu, R. Yu, G. Sun, and Z. Yang. Whistle: Synchronization-free TDOA for localization. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 760–769, 2011.

[188] R. Yamasaki, A. Ogino, T. Tamaki, T. Uta, N. Matsuzawa, and T. Kato. TDOA location system for IEEE 802.11b WLAN. In *Proceedings of the Conference on Wireless Communications and Networking*, pages 2338–2343, 2005.

[189] S. Yang, P. Dessai, M. Verma, and M. Gerla. FreeLoc: Calibration-free crowdsourced indoor localization. In *Proceedings of the International Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 2481–2489, 2013.

[190] J. Ye, L. Coyle, S. Dobson, and P. Nixon. A unified semantics space model. In *Proceedings of the International Symposium on Location and Context-Awareness (LoCA)*, pages 103–120, 2007.

[191] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computer Survey*, 38(4), 2006.

[192] M. Youssef and A. Agrawala. Handling samples correlation in the Horus system. In *Proceedings of the International Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1023–1031, 2004.

[193] M. Youssef and A. Agrawala. The Horus WLAN location determination system. In *Proceedings of the International Conference on Mobile systems, applications, and services (MobiSys)*, pages 205–218, 2005.

[194] M. Youssef, A. Agrawala, and A. U. Shankar. WLAN location determination via clustering and probability distributions. In *Proceedings of the International Conference on Pervasive Computing and Communications (PerCom)*, pages 143–150, 2003.

[195] G. Yu and J.-M. Morel. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[196] Y. Zhao, M. Li, and F. Shi. Indoor radio propagation model based on dominant path. *International Journal of Communications, Network and System Sciences*, 3(3):330–337, 2010.

[197] G. Zhong, I. Goldberg, and U. Hengartner. Louis, Lester and Pierre: Three protocols for location privacy. In *Lecture Notes in Computer Science (LNCS)*, pages 62–76. 2007.

[198] J. Zhu, K. Zeng, K. H. Kim, and P. Mohapatra. Improving crowd-sourced Wi-Fi localization systems using Bluetooth beacons. In *Proceedings of the International Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 290–298, 2012.

University of Murcia
Faculty of Computer Science