UNIVERSIDAD POLITÉCNICA DE
MADRID

ACCURATE INDOOR POSITIONING
AND TRACKING BASED ON DATA
FUSION FROM INERTIAL SENSORS,
WIRELESS SIGNALS AND IMAGE
DEVICES USING DEEP LEARNING
TECHNIQUES

Doctorado en Tecnologías y Sistemas de Comunicaciones

Alberto Belmonte Hernández

Máster Universitario en Ingeniería de Telecomunicación

2019

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN

DEPARTAMENTO DE SEÑALES, SISTEMAS Y RADIOCOMUNICACIONES

**Accurate indoor positioning and tracking
based on data fusion from inertial sensors,
wireless signals and image devices using
Deep Learning techniques**

Author: Alberto Belmonte Hernández - Máster Universitario en
Ingeniería de Telecomunicación

Director: Federico Álvarez García - Doctor en Ingeniería de
Telecomunicación

2019

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

DEPARTAMENTO DE SEÑALES, SISTEMAS Y RADIOCOMUNICACIONES

**Accurate indoor positioning and tracking based on data fusion from
inertial sensors, wireless signals and image devices using Deep Learning
techniques**

Autor: Alberto Belmonte Hernández
Director: Dr. Federico Álvarez García

El tribunal nombrado para juzgar la tesis arriba indicada el día de de
, compuesto de los siguientes doctores:

Presidente: _____

Vocal: _____

Vocal: _____

Vocal: _____

Secretario: _____

Realizado el acto de lectura y defensa de la Tesis doctoral, a de
de , acuerdan la calificación de:

Calificación: _____

EL PRESIDENTE

EL SECRETARIO

LOS VOCALES

UNIVERSIDAD POLITÉCNICA DE
MADRID

ACCURATE INDOOR POSITIONING
AND TRACKING BASED ON DATA
FUSION FROM INERTIAL SENSORS,
WIRELESS SIGNALS AND IMAGE
DEVICES USING DEEP LEARNING
TECHNIQUES

Doctorado en Tecnologías y Sistemas de Comunicaciones

Alberto Belmonte Hernández

Máster Universitario en Ingeniería de Telecomunicación

2019

Summary

Indoor Positioning Systems (IPS) became one of the main interests for researchers in the last years due to the potential number of applications that can be developed in different fields such as health, security, retail, etc. Normally, the proposed solutions try to increase the accuracy of indoor person tracking by using different technological proposals and therefore this is the focus of this doctoral thesis.

Two main ways to solve this problem are, on the one hand, the use of sensory solutions of different nature (wireless, motion, depth...) to get information about the motion of a person or in the other hand the use of image-based devices. Although different solutions have been developed with only one technology, the interest of the combination of several of them has not been exploited enough. In this thesis the use of wireless signals in combination with motion sensors and image-based solutions involving cameras of different kinds has been proposed.

The main reason to combine these technologies is that the deployed Wireless Sensor Network is able to cover a big area and can get measurements if the device is in the wireless range. In contrast, cameras can monitor big areas but with the limitations of indoor environments. If a person is being tracked by the camera and go outside the camera view, sensors can continue collecting measures allowing the person seamless tracking.

Thanks to the big advances in the deep learning field in the last decade, this thesis exploits the use of these algorithms in order to achieve better accuracy than with traditional machine-learning based methods for wireless sensors tracking and classical computer vision methods. The thesis proposes Artificial Neural Networks (ANNs) to be applied to learn from large datasets how to model the behaviour of the signals and sensors in an indoor environment to perform positioning and tracking with wireless signals and motion sensors. Tracking algorithms try to estimate the next position of an object base on previous information. As it is difficult know which is the best model to use, Recurrent Neural Networks (RNNs) have been used to learn a regression model to track our signals. Generative Adversarial Networks (GANs) have been used in this work in combination with a tracking recovery module to generate new measurements when nothing is received during a tracking of a person in order to recover the trajectory in short periods.

Regarding camera devices, deep learning solutions have been employed for object detection, tracking and reidentification algorithms. Normal field of view (NFoV), Fisheye and 360° cameras have been used to get different types of images and test the algorithms. A complete image-based framework have been developed to get the position of the persons in the coordinates system of the monitored environment.

Finally, the fusion module combines the estimations from both independent systems increasing the final obtained accuracy.

Resumen

Los sistemas de posicionamiento en interiores (IPS) son uno de los principales intereses para los investigadores gracias al número de potenciales aplicaciones que pueden ser desarrolladas. Las soluciones propuestas se centran en el seguimiento preciso de personas. Este es el foco de esta tesis.

El problema puede ser resuelto usando sensores de diferente tipo (inalámbricos, movimiento, profundidad...) para obtener información sobre el movimiento de las personas o el uso de dispositivos de imagen. Muchas soluciones usan un sólo tipo de tecnología, pero el gran interés en la combinación de varias ha sido poco explotado. En este trabajo se propone el uso de señales inalámbricas junto con sensores inerciales y cámaras de diferente óptica.

La principal razón de combinar estas tecnologías es que una red inalámbrica de sensores (WSN) es capaz de monitorizar grandes áreas y recibir medidas dentro del rango inalámbrico. En contraste, las cámaras pueden monitorizar grande entornos pero solamente dentro de la sala en la que se encuentran instaladas. Si una persona está bajo el seguimiento de la cámara y deja la sala la red de sensores es capaz de continuar siguiendo al individuo continuamente.

Gracias al auge del aprendizaje profundo, este trabajo se centra en su uso para conseguir mayor precisión que con técnicas precedentes. Las redes neuronales (ANNs) pueden ser aplicadas para modelar el comportamiento de señales y sensores en entornos de interior para realizar tareas de posicionamiento y seguimiento. Los algoritmos de seguimiento filtran las detecciones para ajustar un modelo de movimiento preciso. Debido a la complejidad se ha optado por el uso de redes recurrentes (RNNs) para crear un modelo de regresión que realice el seguimiento. Las redes generativas (GANs) permiten generar nuevos datos basados en entradas concretas. Un módulo para recuperación del seguimiento ha sido desarrollado para situaciones en las que una persona se pierde y vuelve a ser seguida después de periodos cortos.

En imagen, las soluciones deep learning empleadas se centran en detectores de objetos y algoritmos de seguimiento y reidentificación de personas. Cámaras de visión normal, ojo de pez y 360° han sido usadas para obtener diferente tipo de perspectivas con el fin de testear los algoritmos en ellas. Una solución completa para el seguimiento con imagen ha sido desarrollada para obtener las coordenadas reales de una persona en el entorno monitorizado.

Finalmente, un módulo de fusión combina las estimaciones de ambos sistemas independientes incrementando la precisión final.

Keywords

Wireless Sensor Networks (WSN), Inertial sensors, Received Signal Strength Indicator (RSSI), Normal Field of View (NFoV) cameras, Fisheye cameras, 360° cameras, Undistortion, Equirectangular Projection, CubeMap Projection, Camera Calibration, Detection, Positioning, Tracking, Reidentification, Machine Learning, Deep Learning, Kalman Filter, Particle Filter, Object Detection, Fusion Algorithms, Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs).

Palabras clave

Redes inalámbricas de sensores (WSN), Sensores inerciales, Indicador de nivel de señal recibida (RSSI), Cámaras de visión normal (NFoV), Cámaras ojo de pez, Cámaras 360°, Desdistorsión, Proyección Equirectangular, Proyección CubeMap, Calibración de cámara, Detección, Posicionamiento, Seguimiento, Reidentificación, Aprendizaje de máquina, Aprendizaje profundo de máquina, Filtro de Kalman, Filtro de partículas, Detección de objetos, Algoritmos de Fusión, Redes neuronales artificiales (ANNs), Redes neuronales convolucionales (CNNs), Redes neuronales recurrentes (RNNs), Redes neuronales generativas (GANs).

Agradecimientos

No sé si una sección complicada o sencilla debido a quienes han sido todas esas personas implicadas de alguna manera en lograr que este documento y esta tesis doctoral llegue a su fin y contribuir de alguna manera a mi estado mental saludable para ello. Nadie de estas personas ha escrito este documento, pero todas las que mencionaré han contribuido de alguna manera a que yo haya llegado hasta el fin de esta etapa más o menos sano, más o menos vivo, pero sí con una enorme sonrisa y felicidad al haber finalizado.

Aquel año, septiembre de 2015, empezaba el segundo año de máster y las prisas por buscar un trabajo fin de máster empezaron pues de 30 créditos trataba el asunto. Sinceramente, no recuerdo cómo fue pero el tema de sensores y cacharreo me gustaba y de alguna manera u otra acabé en el Grupo de Aplicación de Telecomunicaciones Visuales (GATV). La primera propuesta de TFM no era demasiado interesante, para ser sinceros, pero empecé a trabajar en ello y muy cerca de un chico Colombiano que parecía saber mucho del tema. Tras pocos días, este chico que en cierta medida quedé yo a su cargo, se dio cuenta de que los cacharros me obedecían muy rápidamente y se cercioró de que podía ser que yo, fuera alguien bueno en algo.

Un día, este chico se acercó a preguntar que tenía hecho y desarrollado y creo que en sus ojos vi algo tras la respuesta que le llamó demasiado la atención. En ese momento y sin ningún tapujo me dijo: "A ver, lo que te han propuesto hacer es una mierda, yo tengo ideas interesantes para hacer con este tipo de sensores". En ese momento algo sucedió. El mundo se paró, y un dibujo que jamás olvidaré (ojalá pudiera tenerlo para ponerlo aquí, pero sí añadido un boceto) comenzó el camino que definió tanto mi TFM como el comienzo de esta tesis doctoral, una nueva aventura (quizás la más dura) en la que me he embarcado en mi vida.

Pasó el tiempo y mi TFM llegaba a su fin y algo tuve que hacer bien ya que me ofrecieron continuar en el grupo trabajando en proyectos de investigación y empezando esta tesis doctoral. Y ese fue el comienzo en el grupo de trabajo donde ya van más de 3 años luchando, sudando, riendo, llorando ¿?, pero la palabra que lo define todo es trabajando y bastante duro. Trabajar, constantemente, horas y horas. Durante este tiempo, varios proyectos a la vez europeos y nacionales, dirigir TFG y TFM, empezar a realizar laboratorios en asignaturas y dirigirlos entre otras miles de cosas a las que sumar la que de verdad en la cabeza consumía la gran parte de las ganas de vivir, ésta tesis doctoral para finalmente obtener plaza de profesor ayudante y sumar otra complicación al asunto.

Figure 1: Boceto del principio del fin.

En base a esta historia, comienzo con lo que de verdad es esta sección, esos agradecimientos a quien por poco que sea, ha aportado algo a este trabajo o a mí como persona durante este tiempo en esta aventura. En primer lugar y por ser lo más cercano, gracias al GATV. Para comenzar, gracias a mi jefe en su momento José Manuel, por haber confiado en mí y ver que yo era una persona resolutiva y con ideas nuevas. Todo esto ha llevado a crear un buen ambiente de trabajo y así seguir creciendo no sólo nosotros sino el grupo de investigación. Por supuesto, gracias Federico, tutor durante mucho tiempo tanto del TFM como de esta tesis. Este agradecimiento es especial porque tú has sido en la mayor parte el responsable de estar dónde estoy y de que logre muchas metas en cuanto a trabajo en proyectos a nivel europeo, redacción de artículos, toda la ayuda para lograr el objetivo de ser profesor y, por supuesto, los 100 metros lisos para presentar esta tesis. Dentro de este grupo incluyo a David que pese a no trabajar directamente codo con codo hemos colaborado en alguna situación.

Seguimos con mis compañeros de aventura día a día. Quiero empezar con Silvia, a mi parecer la cabeza que nos tiene cuerdos en este descontrol. Trabajadora incansable, trato personal envidiable, gran madre de su fauna personal y de la fauna del GATV y con una sonrisa que quieres ver siempre en su cara si trabajas con ella directamente.

Continuamos con Fran, Franman, el superhéroe. Si no hay solución para algo sólo tienes que pensar que alguien ya lo habrá hecho. El troleo en persona, pero un maldito genio al que envidiamos en su trabajo y proyectos. Pese a que quiso dejarnos de lado para adentrarse en su aventura personal (hijos, dos, en muy poco tiempo) siempre está ahí para lo que haga falta.

David, el cabeza centrada, el que no para, su trabajo es impecable, hemos aprendido tanto yo de él como él de mí. Quizás somos los programadores de Inteligencia Artificial principales, con muchas ideas y con ganas de hacer cosas increíbles. Pero, además, una persona como ninguna otra, con sus rarezas y sus virtudes pero es lo que nos hace a todos especiales.

Alvaro, o ya por culpa de una voz cantante, Alvarito. La tranquilidad, la serenidad, el saber hacer. Si la calidad de lo que ves es un desastre te contará hasta el número de bloques de 8x8 que aparecen en la imagen. Sería algo así como todo lo bonito concentrado en un ser. No cambies nunca ser de luz.

Una mención a Giussepe, el manitas del grupo, el cirujano de la electrónica. Una voz en off en italiano maldiciendo al mundo de vez en cuando si algo se resiste. Juanan, para mí un desconocido al que conozco. No hemos trabajado juntos pero las pocas veces que nos hemos podido juntar para tomar algo son para recordar. Chus, como empezar. Nuestro ángel de la gestión y la voz del no estar de acuerdo nunca en todo lo que pensamos. Aun así otra compañera del grupo que nos resuelve una serie de problemas que escapan a nuestro alcance, gracias. Miguel, de las últimas incorporaciones que pese a no tener tanta relación nos ha enseñado un mundo que no conocíamos del todo. Espero más adelante trabajar en algo conjuntamente. Javi, compañero con el que he tenido menos relación pero que al menos en muchos gustos coincidimos ya sean frikis o de música. Jorge, el señor de la sonrisa, siempre alegre y transmitiendo buenas sensaciones, sigue así contigo mismo y con todos nosotros. Y Andrés, Andresito, el inconforme con la vida, la representación del malestar. Nada es adecuado, nada es lo correcto. Una persona joven, con ideas diferentes y formas de ser complicadas de entender para los "mayores". Una sonrisa conseguida todos los días.

Y finalmente, todos los compañeros ya lo saben, podría llamarlo mi hermano, mi extensión, mi mano derecha (soy zurdo y él también no le daremos ese privilegio). Gustavo (el Colombiano), el culpable principalmente de dónde me encuentro, de dónde estoy y dónde he llegado gracias a todo lo que aprendí de él. Mi maestro, mi amigo y lo que vaya más allá de mejor amigo. Necesitaría páginas y páginas para agradecerle en primer lugar todo lo que hiciste conmigo en el comienzo, y otras cuantas para seguir narrando nuestras aventuras ya sean de trabajo o de vida. No quiero extenderme y prefiero decirte a la cara, que es donde de verdad se muestra la realidad, todo lo que te aprecio y todo lo que te quiero, de corazón.

Seré breve pero quiero incluir a otra serie de personas en estos agradecimientos. En primer lugar, mis amigos, tanto de Madrid como de Murcia. Por parte de los madrileños, gracias por estar ahí, por aguantarme, puesto que a veces he sido un pesado y por entender a veces mi agobio y preocupaciones y pagarlo con vosotros más de una vez cuando sólo habéis estado ahí

intentado ayudarme de alguna manera. En cuanto a los murcianos (Doloreños), simplemente mis amigos de toda la vida, los que me sacan la sonrisa cuando estoy allí, los que alucinan cuando les cuento los proyectos, las cosas que hago, que me prestan atención y que, además, después de tantos y tantos años me quieren y les quiero infinito.

Entre estos otros agradecimientos, dos personas que han sido bastante especiales para mí que han aprendido conmigo, así como yo he aprendido con ellas. Clara y Lucía, TFG y TFM (porque parece que algo hice bien) y TFM respectivamente. Gracias a ellas me enamoré de verdad de la enseñanza, aunque ésta ya me encantara, gracias a su trabajo incansable, ganas de aprender y de hacer más y más. Os debo mucho y ojalá todos mis futuros trabajos con alumnos puedan ser tan divertidos y complicados a la vez cómo con vosotras dos.

Para acabar, por supuesto, mencionar a mi familia. Por una parte a mis abuelos, a aquellos que no han podido estar para ver este momento pero que han sido una parte de mi vida complicada por parte de toda la familia implicada pero que me dieron todo su amor en todo momento y que recordaré a ambos por siempre, y a los otros abuelos que me quieren que no paran de hablar de mí, y que me dan ánimos siempre para seguir adelante y conseguir todo lo que me proponga. A mis tíos ahí presentes también animando y que estarán ahí para cualquier problema.

Y por supuesto, lo mejor para el final. A mi familia directa, a mis padres y mi hermano. No han contribuido en el desarrollo directo de mis experimentos, de la escritura de esta tesis, de la escritura de los artículos publicados, pero son ese público expectante constantemente y preocupado porque yo siempre esté bien, no lo pase mal y de darme y ofrecerme todo lo necesario para estar feliz pase lo que pase. Suelo decirles algo que a veces surge en ciertas conversaciones y es que yo voy a ser inmortal. Ellos se lo toman a broma y pues acabamos divagando y que me vaya olvidando de ello. Lo que ellos no saben es que hay muchas formas de ser inmortal. A parte de la de vivir para siempre, para mí otra manera de ser inmortal es que haya al menos una sola persona que te quiera y te respete para siempre. Siguiendo esta premisa, yo ya sé que voy a ser inmortal, pero inmortal por triplicado, gracias a mi madre, gracias a mi padre y gracias a mi hermano, os quiero.

Gracias a todos, Alberto Belmonte Hernández.

Index

1	Introduction	1
2	Indoor Positioning Systems (IPS)	5
2.1	Technologies employed in IPS	9
2.1.1	Wireless Based Technologies used in IPS	9
2.1.2	Inertial Navigation Systems (INS) and Pedestrian Dead Reckoning (PDR)	14
2.1.3	Image Based Technologies used in IPS	16
3	Machine and Deep Learning	24
3.1	Machine learning	25
3.2	Deep Learning	32
4	Overview of the complete proposed system architecture	41
5	Proposed Wireless and Inertial IPS solutions	46

5.1	State of the art IPS solutions without learning algorithms using Wireless and Inertial sensors	46
5.1.1	Proximity Detection (Connectivity Based Positioning)	46
5.1.2	Triangulation	47
5.1.3	Angle Based Method	47
5.1.4	Time Based Methods	48
5.1.5	Signal Property Based Method	49
5.1.6	Dead Reckoning (DR)	49
5.1.7	Map Matching (MM)	50
5.1.8	Fingerprinting Based Indoor Localization	50
5.1.9	IPS combining Positioning and Tracking	51
5.2	State of the art IPS solutions using learning algorithms and Wireless and Inertial sensors	52
5.2.1	Machine/Deep Learning using Wireless Signals	52
5.2.2	Fingerprinting recovering and update	55
5.2.3	Machine/Deep Learning in INS and Dead Reckoning	57
5.3	Proposed sensors system architectures	60
5.3.1	Fingerprinting definition	60
5.3.2	Fingerprint calculation	63
5.3.3	Probabilistic sensor fusion architecture	66
5.3.4	ANN sensor fusion architecture	83
5.3.5	Recurrent Neural Networks for continuous indoor tracking	102

5.3.6	Tracking recovery module	108
5.4	Research contributions for indoor tracking with wireless and inertial sensors . .	113
6	Proposed image based IPS solutions	114
6.1	State of the art IPS solutions without learning algorithms using Image devices	114
6.1.1	Foreground-Background Segmentation in ISS	114
6.1.2	Object Tracking and Re-Identification in ISS	117
6.1.3	Behavioral Analysis	118
6.2	State of the art IPS solutions using learning algorithms and Image devices . . .	120
6.2.1	Machine/Deep Learning for person detection and tracking using image devices	120
6.2.2	Machine/Deep Learning for person re-identification using image devices	124
6.3	Proposed Cameras Architecture	128
6.4	Image projections	130
6.5	Camera calibration	139
6.6	Image person detection	145
6.6.1	Object detection algorithms	146
6.7	Person reidentification	150
6.8	Image person tracking	157
6.8.1	Visual online tracking using image information	158
6.8.2	Visual tracking with Deep Learning algorithms	160
6.8.3	Mixing detection, tracking and reidentification	163

6.9	Image detection, reidentification and tracking results	165
6.9.1	Test dataset	166
6.9.2	Results on Deep Learning objects detection algorithms	169
6.9.3	Results on Deep Learning reidentification algorithms	175
6.9.4	Results on Visual (no learning)/Deep Learning tracking algorithms . . .	179
6.10	Proposed framework for Detection, Tracking and Reidentification (DTR) . . .	182
6.11	Research contributions for indoor tracking with cameras of different field of view	187
7	Multisensory IPS Fusion	188
7.1	Person Identification and Multi-modal Fusion	188
7.2	Data synchronization	189
7.3	Track-to-Track and Orientation comparison	191
7.4	Multimodal trajectory fusion	193
7.5	Research contributions in the final fusion of trajectories extracted from different systems	195
8	Real applications using the proposed system	196
8.1	Patient tracking and monitoring for local and remote rehabilitation at home . .	196
8.1.1	Proposed solution for Patients monitoring	197
8.1.2	Results of the system in a real scenario with patients.	199
8.2	Automatic detection and tracking of characters in the scene for accessibility services	201
8.2.1	Proposed accessibility services fot the TV	202

8.2.2	Proposed approach for accessibility services using Deep Learning techniques	203
9	Conclusions and future lines	209
9.1	Conclusions	209
9.2	Future lines	212
	References	214

Index of figures

2.1	<i>Left:</i> IPS Market in USD by application. <i>Right:</i> IPS Market in USD by technology.	6
2.2	Typical RFID architecture. Several antennas are deployed connected typically to one RFID reader that is able to work with more than tens of them. RFID tags are allocated in moving objects or persons and work passively through the monitored environment.	10
2.3	WLAN indoor positioning architecture. Access points are deployed covering big indoor areas and are connected to the same network to share information between them and a global server with all the positioning methods.	11
2.4	Wireless Sensor Networks advantages.	12
2.5	UWB spectrum compared with narrow-band and spread-spectrum technologies.	13
2.6	IPS Infrared Architecture using LEDs information. A device needs a sensor able to collect infrared information sent by the main architecture. The receiver processes the information to provide the actual position.	13
2.7	Typical Inertial Navigation System. Accelerometer and Gyroscope measurements can be combined to estimate a lot of other important parameter to get position and orientation of a moving object. Initial conditions are necessary to start the system with the correct parameters.	14

2.8	Typical Pedestrian Dead Reckoning Architecture. All the information is extracted from sensors and a sensor fusion and a step detection algorithm are combining to get all the necessary parameters to know person position and orientation.	15
2.9	Left and center: Examples of CCTV cameras used in surveillance systems. Right: Typical CCTV cameras installation for surveillance and traffic control. .	17
2.10	Camera focal length depending on aperture angle (FoV) in CCTV cameras. . .	18
2.11	Example of Fisheye images. Top left: Hemispheric image from a ceiling camera. Top right: N FoV projection of 65° over a part of the image. Bottom: Panoramic projection from the Hemispheric image (loss of information in the center of the image)	19
2.12	Left: Kinect v1 camera (infrared speckle pattern). Center: MYNT Eye 3D (stereo depth camera sensor) [1]. Right: Kinect v2 camera (Time of Flight measurement.)	20
2.13	Example of quotidian actions recorded with Kinect v2 camera. Top part represents the RGB image collected by the camera. The bottom part contains the Depth image extracted with the camera and the estimated skeletons of the persons (Kinect v2 camera intrinsic feature) presented in the image.	21
2.14	Most common 360° camera projections. From left to right: Two Fisheye 180° projections, Equirectangular projection (rectangular image that contains all the 360° information), Domemaster (representation in e 3D sphere) and CubeMap (6 N FoV projections in a cube).	22
2.15	Different 360° camera types. From left to right: Rig camera (combination of 6 cameras) [2], mosaic system (one normal camera rotates along several angles stitching images), two fisheye 180° lens joined together with a stitching post-processing [3] and camera with more than two spherical lenses [4].	22
3.1	Definitions of what is machine learning by different known companies and universities.	25

3.2	Machine Learning project chain starting from data acquisition and cleaning, passing through the learning phase and finalizing with the model deployment. .	26
3.3	Machine Learning techniques grouped by style and functions.	27
3.4	Common Metrics for different types of Machine Learning problems.	31
3.5	Scheme for Feature Selection, Transformation and Hyperparameters Tuning in a Machine Learning problem.	32
3.6	Top-5 error on ImageNet Dataset (bigger world dataset of images). The accuracy in 2015 using Deep Learning networks equal or improve the accuracy presented by human at classification level.	33
3.7	Machine Learning manual feature extraction against Deep Learning automatic feature extraction working graph.	34
3.8	General architecture of a Fully Connected Deep Neural Network.	36
3.9	General architecture of a CNN network for classification tasks.	37
3.10	General architecture of a RNN network. Sequential information is given as input and is feed into the network in different timesteps.	38
3.11	General architecture of an Autoencoder where encoder/decoder parts are presented. In this example the autoencoder try to reproduce the input at the output having in the middle a compress representation of the data.	38
3.12	General architecture of a Generative Adversarial Network (GAN) with the Generator (G) and the Discriminator (D) that play together in the Loss function of a GAN.	39
3.13	Deep Reinforce Learning conceptual system diagram architecture.	40
4.1	Proposed complete system architecture to fuse both wireless/inertial architecture and camera person tracking architecture.	42
4.2	<i>Left:</i> Raspberry Pi 3 Model B. <i>Right:</i> Raspberry Pi Zero	44

4.3	Wireless devices and IMU+Magnetometer used in this work. From left to right: XBee S1, Bluetooth USB Dongle PBU40, WiFi USB dongle TP-LINK TL-WN722N, Adafruit 9-DOF LSM9DS0.	44
4.4	Selected cameras in this work. <i>Left</i> : Vivotek FD8179-HV (NFoV camera). <i>Middle</i> : MOBOTIX c25 Hemispheric Indoor Camera (180° camera). <i>Right</i> : RICOH THETA V (360° camera).	44
4.5	Scenario where all the experiments were performed. Allocation of all the devices is drawn and approximated sizes of each room and corridors is presented.	45
5.1	Example of trilateration method where 3 nodes (Raspberry Pi as RPI) are used to estimate 3 distances (r1, r2, r3) and the intersection between the circles using and optimization algorithm provide the final estimated position P.	47
5.2	Calculation of 2D position based on two antennas using AoA technique.	48
5.3	Dead reckoning example where step length is estimated along bearing angle to get the next position.	49
5.4	Fingerprinting procedure, phase 1 (fingerprinting generation and characterization) and phase 2 (new measurements matching)	51
5.5	<i>Left</i> : Example of an Autoencoder network to learn to compress the information (RSSI from several nodes as input) and after use the encoded information to learn to predict the position. <i>Right</i> : Example of Deep Neural Network with an initial Encoder part to compress the information and a Classification/Regression part to predict location, building or floor depending on the available information.	53
5.6	AF-DCGAN architecture [5]. It is composed by a Generative Adversarial Network which is able to generate Channel State Indicator (CSI) maps in indoor environments.	56
5.7	Deepsense architecture [6]. It is composed by a combination of Convolutional Neural Networks and Recurrent Neural Network to work with data in both temporal and spatial domain.	59

5.8	<i>Left:</i> This figure represents the real monitored scenario and the cells distribution for fingerprinting. Five nodes equipped with the selected wireless technologies are deployed across the area (<i>circles</i>). Doors are drawn in red and furniture in gray respectively. <i>Right:</i> the reference coordinates system for inertial and magnetic sensors as well as the allocation of devices employed are illustrated.	61
5.9	Example of RSSI variance boxplots in the collected Fingerprinting database is presented for each node in different cells selected for a particular wireless technology (XBee). Variance in measurements is large, yielding to significant fluctuations in the signal distribution over a particular location (i.e. a cell in the monitored room).	62
5.10	An example of different orientations in the same cell is presented. The orientations have been concentrated in four angle ranges. The large variability in the measurements depends heavily on the orientation of the device in the cell.	65
5.11	Kalman Filter Algorithm steps for prediction and estimation.	70
5.12	<i>Track-to-track Fusion Architecture:</i> Separate processing stages for each sensor to achieve high-level inferences that are subsequently fused.	74
5.13	<i>Kalman Sensor Group Fusion Architecture:</i> direct sensor data combination.	76
5.14	Cell splitting for experiments using the proposed probabilistic architecture.	79
5.15	Average error comparison of route estimation for every technology (no fusion applied) using Fingerprinting technique, Kernel Density Estimation and Kalman Filter.	80
5.16	Average error using both fusion architectures in path estimation for an experiment composed by 300 routes with 90 steps per route.	81
5.17	<i>(left)</i> Route estimation by using Track-To-Track Fusion Architecture and <i>(right)</i> route estimation by using Kalman Sensor Group Fusion Architecture.	82

- 5.18 SwiBlux system architecture is composed by several stages: *First*, a synchronization process and a Deep Learning approach is used to perform the fingerprinting estimation (green boxes). *Second*, a Gaussian Filter is proposed to reduce outliers in the estimations (orange boxes). *Third*, the step, velocity estimation and absolute orientation (blue boxes). *Fourth*, Particle Filter tracks the person based on a realistic movement model giving the final position (yellow boxes). 85
- 5.19 Deep Neural Network composed of: an input layer with size the selected feature vector, two hidden layers and the last layer with size equal to the number of cells with a final Softmax activation. 86
- 5.20 General Scheme of the Training process. *From top to bottom*: Fingerprint creation from the N Network nodes for each particular technology in addition to the orientation values (Yaw) is depicted. The length of the feature vector inserted into the Neural Network is $(I(nodes) \times \lambda(technologies)) + 1(Yaw)$. . . 87
- 5.21 Example of the Gaussian Outliers Filtering. The bidimensional Gaussian is centered at the previous estimated position and the deviation is the velocity calculated at the current stage. The cell estimated by the Neural Network, in this case, is wrong so the filter applies a low probability to it. In this manner, the larger probability is assigned to the nearest cells (the ones that the filter covers). 89
- 5.22 *Left*: Three axis accelerometer measurements. Regarding the proposed referenced coordinate system it can be shown that the Y axis measurements are around $1g$ representing the gravity force ($9.8m/s^2$) in this axis. *Center*: Example of peaks/valleys detection. Red dots represent valleys and green dots represent peaks. It can be observed that the amplitude needs to be large between peaks/valleys to detect them correctly. *Right*: Estimated velocity after Monte Carlo Simulation to get the parameters for the movement model and filtering the output. 91
- 5.23 Magnetometer calibration results. *Left*: uncalibrated normalized data points, *right*: calibrated normalized data points after ellipsoid calibration. 93

5.24	Estimation accuracy in position estimation using several methods with different feature vectors in two main cases: The first case (<i>left</i>) corresponds to perform the normal Fingerprinting estimation based on the selected method, the second (<i>right</i>) shows the results after applying the GOF and the weighted combination proposed in this paper. For horizontal axis X, B and W denote XBee, Bluetooth and WiFi technologies respectively.	97
5.25	Monte Carlo simulation over a range of possible values for adjust the the step length model to the detected peaks and valleys by the adaptive step jerk algorithm.	98
5.26	Particles (blue dots) managed by the Particle Filter at different moments during the tracking. The red dots represent the estimated position at each moment based on the particles whereas blue line depicts the final path estimated using the proposed architecture.	100
5.27	Training Loss and Accuracy for Training and Validation sets in two hidden layers (256, 64 neurons respectively) neural network configuration.	101
5.28	Recurrent Neural Network system architecture. It is composed by the WSN that collect RSSI measurements and the velocity and orientation estimators. The recurrent neural network predicts next estimated position using the proposed feature vector. Tracking recovery is applied when no measurements are received and use a Conditional GAN that generates RSSI maps and a second Recurrent Neural Network with a different input.	104
5.29	CDF error tracking in the routes validation set for different RNN configurations and the proposed modified loss function with a regularization parameter. . . .	107
5.30	Error increment after different periods with the system working for both RNN with feature vector (FV1) and sequence length 15 and RNN with feature vector (FV2) and sequence length 15.	108

5.31	Conditional Generative Adversarial Network architecture. The input is a vector of Gaussian noise and the proposed one hot labels. The output of the Generator part after training is the RSSI maps. The Discriminator part help to learn which images are Fake or Real. Numbers represents in case of convolutions the number of filters employed and, in the neurons layers, the number of neurons.	109
5.32	Example of original created RSSI maps in the proposed room. <i>Top</i> : Real RSSI maps in different times of the day. <i>Bottom</i> : Generated RSSI maps using the proposed Conditional GAN in different day hours.	111
6.1	Example of two simple algorithms for background segmentation in pedestrian and pedestrian and vehicles environments. Left: Mixture of Gaussians (MoG algorithm). Right: Median Blur algorithm.	115
6.2	Most extended person detection chain algorithm using HOG and SVMs. The first step use background segmentation algorithms to extract possible object in movements. HOG descriptor and SVM classifier is applied over the detection to get if is a person or not.	116
6.3	Left: Box plot illustrating averages and distributions of frame rates. Right: Success plots of the five tested trackers showing the fraction of frames successfully tracked as a function of required Jaccard index. Results presented in [7].	118
6.4	Detection and people tracking is performed to get information about person movement in time. Next steps presents feature extraction like "Key-points". Finally, using this information the estimated behavior is obtained.	119
6.5	Deep Learning object detector algorithms in time. In red the most known algorithms are marked.	120
6.6	Region Convolutional Neural Network (RCNN) family.	121
6.7	Keypoints human detection algorithms. Left: OpenPose (Skeleton, hands and face detection). Center: AlphaPose (Real time detection in crowd areas). Right: DensePose (Estimated 3D mesh).	122

6.8	Deep 360 Piloting network architecture. Object detection and Recurrent operator to learn in time to track more salient actions.	123
6.9	Most extended Re-Identification methods.	126
6.10	Proposed Cameras Architecture for person detection, tracking and reidentification.	128
6.11	Example of the images collected by each type of the proposed cameras. <i>Top left</i> : NFoV camera. <i>Top Right</i> : Fisheye camera. <i>Bottom</i> : 360° camera.	130
6.12	Fisheye unrolling process to map points in the input image to a pseudo panoramic view.	131
6.13	Examples of Fisheye and unrolled images.	133
6.14	<i>Left</i> : Equirectangular projection of a 360° images. <i>Right</i> : Cubemap projection obtained from the Equirectangular image.	134
6.15	Intuition of the sphere to plane projection and example of NFoV image extracted from the equirectangular image.	134
6.16	Visual explanation of Equirectangular/Fisheye to Fisheye/Equirectangular projection respectively.	137
6.17	Example of equirectangular image converted to dual fisheye with 180° of aperture in both sides.	138
6.18	Examples of same equirectangular image with the camera rotated in different axis.	139
6.19	Pinhole camera model.	140
6.20	<i>Left</i> : Image with no distortions. <i>Center</i> : Image with Radial distortion. <i>Right</i> : Image with Tangential distortion	142
6.21	Example of NFoV images taken with the chessboard pattern in different positions of the image for calibration.	143

6.22	Example of Fisheye images taken with the chessboard pattern in different positions of the image for calibration.	143
6.23	NFoV Undistorted	144
6.24	Zenith Undistort	144
6.25	Process to convert between pixel coordinates to real world coordinates. <i>Left:</i> Original image, <i>Center:</i> Undistorted image, <i>Right:</i> Selected points in the image for calibration.	145
6.26	YOLOv3 network architecture.	146
6.27	Mask-RCNN network architecture.	147
6.28	Single Shot Detector network architecture.	148
6.29	RetinaNet network architecture.	149
6.30	Region Fully Convolutional Network architecture.	150
6.31	Example of images in different reidentification datasets.	153
6.32	<i>Left:</i> MuDeep algorithm intuition. <i>Right:</i> MuDeep multiscale network architecture	154
6.33	HACNN network architecture where attention modules are computed after Inception networks blocks and shared between different image representations along the network.	155
6.34	PCB network architecture where the uniform partitioning is showed.	155
6.35	MLFN network architecture where multiple blocks extract independent features that are combined to be the input of more deeper layers.	156
6.36	GOTURN tracker architecture.	160
6.37	Architecture of the Fully-Convolutional Siamese Network for object tracking. .	161
6.38	Architecture of the Re3 network for object tracking.	162

6.39	Deep-SORT architecture composed by an initial person detector and a multi tracking algorithm using Kalman filter with a deep association metric.	163
6.40	Detect and Track network architecture for object detection and tracking in a single framework.	164
6.41	Alpha Pose architecture with spatial transformer and SPPE network for body keypoint detection.	165
6.42	Example of an equirectangular image and the extended version to avoid person cut in borders.	167
6.43	Example of test annotation in the different images used for all the camera types. <i>Top</i> : NFov and undistorted NFov images. <i>Center</i> : Unwrapped Fisheye and Fisheye images. <i>Bottom</i> : 360° Cubemap and 360 equirectangular images. .	168
6.44	Bounding box accuracy obtained by several deep learning object detector algorithms pretrained on COCO dataset.	171
6.45	Detection accuracy obtained by several deep learning object detector algorithms pretrained on COCO dataset.	173
6.46	Processing time obtained by several deep learning object detector algorithms pretrained on COCO dataset.	174
6.47	Example of detections made by Yolo v3 algorithms in different image types. . .	175
6.48	Example of query images and RANK3 matches for different persons.	178
6.49	Example of tracking occlusions and tracking ID change in three types of images.	182
6.50	Complete workflow of the final proposed framework for detection, tracking and reidentification.	184
7.1	Example of data collected in time with different data rates.	189
7.2	Sensors data inside predefined windows time.	190
7.3	Data after synchronization.	191

7.4	Example of trajectories from all the devices involved in different time moments.	192
7.5	Example of a part of a trajectory yaw from the Sensors Device and the Fisheye camera for the same person.	193
8.1	Deployed system in a home composed by a WSN, a Fisheye camera and two Kinect cameras.	198
8.2	Example of images taken by the deployed system in a home. <i>Left</i> : Image from the Kinect camera installed in the living room. <i>Right</i> : Image collected by the Fisheye camera mounted at the ceiling.	199
8.3	Evolution of patient location within home over a 24-hour period. The time the patient is outside is not monitored by the system.	201
8.4	Example of automatic face magnification in a preprocessed content.	202
8.5	Example of information extracted from faces showed in the video.	203
8.6	Complete framework for face detection, tracking, recognition and characterization.	204
8.7	Examples of the main characters (top) and some images in our dataset for some of them (bottom).	206
8.8	Example of detected faces in video.	207
8.9	Performance comparison between the face detector applied in all frames and the tracking over the first detected face.	208

Glossary

AoA - Angle of Arrival

ToA - Time of Arrival

TDoA - Time Difference of Arrival

E-OTD - Enhanced Observed Time Difference

CSI - Channel State Indicator

RSSI - Received Signal Strength Indicator

WSN - Wireless Sensor Network

LIDAR - Light Detection and Ranging

RADAR - RAdio Detection And Ranging

IPS - Indoor Positioning System

PDR - Pedestrian Dead Reckoning

IR - Infrared

UWB - Ultra Wide Band

WLAN - Wireless Local Area Network

RFID - Radio Frequency Identification

NFC - Near-Field Communication

AI - Artificial Intelligent

ML - Machine Learning

DL - Deep Learning

WMS - Warehouse Management System

AR - Augmented Reality

SoC - System on Chip

CDMA - Code Division Multiple Access

GSM - Global System for Mobile communications

IMU - Inertial Measurement Unit

GNSS - Global Navigation Satellite System

ISS - Intelligent Surveillance System

FoV - Field of View

NFov - Normal Field of View

CCTV - Closed Circuit Television

P2P - Point to Point

KNN - K-Nearest Neighbors

SVM - Support Vector Machine

TRPO - Trust Region Policy Optimization

PPO - Proximal Policy Optimization

A3C - Advantage Actor-Critic

TP - True Positives

TN - True Negatives

FP - False Positives

FN - False Negatives

AUC - Area Under Curve

MAE - Mean Absolute Error

MSE - Minimum Square Error

RL - Reinforcement Learning

DRL - Deep Reinforcement Learning

DNN - Deep Neural Network

ANN - Artificial Neural Network

CNN - Convolutional Neural Network

RNN - Recurrent Neural Network

GAN - Generative Adversarial Network

AE - Auto-Encoder

VAE - Variational Auto-Encoder

RBM - Restricted Boltzman Machine

FC - Fully Connected

P2MP - Point to Multipoint

CoO - Cell of Origin

RToF - Roundtrip Time of Flight

LoB - Line of Bearing

AP - Access Point

LDA - Linear Discriminative Analysis

SVR - Support Vector Regressor

LoS - Line of Sight

SDA - Stacked Denoising Auto-encoder

RMS - Root Mean Square

HMM - Hidden Markov Model

KDE - Kernel Density Estimation

PDF - Probability Distribution Function

MMSE - Minimum Mean Square Error

KF - Kalman Filter

PF - Particle Filter

MLE - Maximum Likelihood Estimator

SGD - Stochastic Gradient Descent

PDE - Probability Density Estimation

LSTM - Long Short Term Memory

OtO - One to One

MtM - Many to Many

CFD - Cumulative Distribution Function

MoG - Mixture of Gaussians

LBP - Local Binary Patterns

HoG - Histogram of Oriented Gradients

CSRT - Discriminative Correlation Filter with Channel and Spatial Reliability

KCF - Kernelized Correlation Filters

TLD - Tracking, Learning and Detection

MOSSE - Minimum Output Sum of Squared Error

RCNN - Region-Based Convolutional Neural Network

SSD - Single Shot Detector

YOLO - You Only Look One

RFCN - Region-Based Fully Convolutional Network

FPN - Feature Pyramid Network

MAP - Mean Average Precision

GPU - Graphics Processing Unit

ResNet - Residual Network

RoI - Region of Interest

Chapter 1

Introduction

In the recent years, scientific methods for data analysis of human activities have become a relevant topic to extract behavioral patterns which represent valuable metrics for multiple purposes such as research, business, security, health, among others. Indoor/outdoor tracking has attracted an extensive focus of research due to the continuous progress in computing capabilities and sensing technologies such as cameras, light detection and ranging (LIDAR), ultrasound, infra-red, etc.

An extended approach for people tracking is based on Computer Vision techniques using RGB sensors. However, occlusions, lighting changes among others, constrain the estimation accuracy of these methods. Moreover, recent RGB-Depth and LIDAR sensors (i.e Intel Real Sense or Microsoft Kinect) allow to obtain not only the RGB image also depth information attaining a significant accuracy improvement compared to RGB estimation. Nonetheless, short detection range in addition to aforementioned issues decrease the application of these devices. There exist several works that employ ultrasound sensors for tracking, however, lack of accuracy, difficulties associated to multi-object detection and coverage range as well, make these approaches unfeasible for scenarios where individualization is required (i.e places with limited capacity).

Mobile technologies have fostered the development of methods for position estimation and tracking due to three main reasons: *(a)* these devices are widely used among population (smartphones), *(b)* fulfill most of the aspects concerning privacy and *(c)* are provided by most of the sensors used for these purposes.

Advances have empowered personal devices with several wireless technologies (IEEE802.15.1 Bluetooth, IEEE802.11g/n WiFi or IEEE802.15.4) that can be used for indoor position estimation. In particular, two metrics are used for the mentioned technologies: Time-of-Arrival (ToA) and Received Signal Strength Indicator (RSSI). The former relies on obtaining the time lag of a packet forwarded to an end-point and estimate the distance between devices. However, strong requirements in terms of hardware and synchronization make this proposal hard to implement. The latter approach is most extended as devices are able to obtain the RSSI parameter from packets although propagation effects such as multipath, fading, reflections among others rise a challenge for processing techniques applied.

The inclusion of inertial sensors (accelerometers, gyroscopes and magnetometers) has increased the set of features that can be applied to prediction models, yielding to meaningful accuracy improvements for person movement monitoring. For the aforementioned techniques (RSSI and ToA), a Wireless Sensor Network (WSN) composed of a set of nodes deployed across the area under monitoring is utilized due to its low-consumption, self-organizing and easy deployment capabilities. These proposals are mainly classified into deterministic or statistical methods. On the one hand, deterministic techniques adopt a theoretical model for propagation and geometrically combine the measurements gathered from several nodes to estimate the person position. On the other hand, statistical techniques, such as *Fingerprinting*, characterize the environment by creating a background dataset that contains the measurements at predefined locations. Several Fingerprinting parameters must be adjusted to achieve good performance in position estimation procedure. However, in most of the existing research, there is not a clear description neither of how these parameters were obtained nor how datasets are updated to face with changes in the scenario.

There exists a very huge effort in the research related to image processing and computer vision. Lots of algorithms have been proposed for several purposes like feature extraction, segmentation, detection, etc. Regarding the purposes of this work, an extensive area of investigation is focused on people detection, tracking and recognition. Traditional methods exploit image characteristics in order to extract useful information from RGB data able to characterize objects and locate them in the image frame.

With the grow of computer capabilities these algorithms have been evolving using more complex ideas that requires more powerful computer or server dedicated only for processing. Feature engineering is the most extended topic and consist on the application of algorithms over the image to get vector of numbers that can define unequivocally objects, actions, or interesting parts of the image to perform in a second part some distance metric able to categorize

the extracted feature vectors.

The decreasing in price have empowered the use of more complex types of cameras that provide an extra information that facilitate or add more useful data for the algorithms. In this context the most common types are RGB-D and Omnidirectional Lens cameras. The first type provides in conjunction with RGB image a depth image that is used to extract information in the 3D space. The seconds relies on the use of different lens able to get information in very big areas with only one camera. Fisheye cameras provide a panoramic view of 180° and usually are allocated on the wall or in the ceiling. In the other hand, 360° cameras provide an image that cover the entire view around the camera.

Currently, Machine and Deep Learning algorithms provide novel tools to cope with large datasets, learning complex non-linear patterns from input information. Topics involve data science, image classification, detection and recognition, signal processing, among others. In addition, the huge number of public datasets containing data of different nature have increase quickly the extensive use of these algorithms.

Regarding the proposed scenario, Fingerprinting can be treated as a huge dataset of RSSI measurements where these algorithms can be applied. Regarding image devices, algorithms can learn automatically from input feature vectors (machine learning) or learn directly more complex hidden patterns automatically using Deep Learning techniques. In this case, there exists lots of very complete datasets that can be used in the scope of this work.

The main proposal is to develop a complete framework for indoor positioning and tracking using wireless technologies, inertial sensors and different types of cameras. The main contributions to the research community are the following:

- A complete framework for indoor tracking using probabilistic methods and fingerprinting technique to estimate position and fusion algorithms to combine estimations from several wireless technologies.
- A complete architecture for accurate indoor tracking based on a Deep Learning Fingerprinting and particle filters in a combination of wireless signals and inertial sensors improving the accuracy of the previous proposal.
- An architecture using Recurrent Neural Networks to learn in time to predict and track the position using wireless signals and inertial sensors.

- A method to recover measurements when signals are lost and the person is inside the covering range using Generative Adversarial Networks (GANs) and Recurrent Neural Networks (RNNs) to work in time.
- A complete architecture for image based indoor tracking combining deep learning algorithms for detection, tracking and re-identification using different camera types (NFoV, Fisheye and 360° cameras).
- The complete framework that combines both approaches for sensors and cameras and a late fusion combination to get the final tracked position.
- Real world applications of the proposed methods in different topics.

The remainder of this work is organized as follows: chapter 2 present an introduction to Indoor Positioning Systems (IPS). In chapter 3, the overall machine and deep learning solutions are drawn. Chapter 4 details the complete proposed framework that combine sensors and cameras for accurate indoor tracking. The proposed methods using wireless signals and inertial sensors are outlined in chapter 5. Methods for image-based solution are described in chapter 6. Results of the complete proposed framework are collected in chapter 7. Chapter 8 collect some of the proposed method applied to different real work solutions. Finally, in chapter 9 the main conclusions and future work are drawn.

Chapter 2

Indoor Positioning Systems (IPS)

Indoor positioning systems (IPS) are one of the most important fields of study in terms of provide accurate position estimations in areas where other common extended systems doesn't work properly. IPS are very attractive to the scientific and enterprise community due to the numerous applications and the big market opportunity that exist applying these technologies [8, 9]. It is observed the exponential growth not only in the applications but also in the technologies used (Figure 2.1).

Based on this success, the next question is, what is IPS?. One first definition can be: *"Indoor positioning systems enable locating the position of objects or people within buildings"*. The main word in this sentence is *"position"* and the second part relies on the concept of *"buildings"*. Then based on this premises, IPS is a system able to provide information about the location in indoor environments. But how? This definition is very simple and some important information is missing. A second definition can be the following: *IPS is a system of network connected devices which is used for sensors positioning of objects and persons inside buildings (indoor environments) and partly covered areas"*. Now important information is presented about what requirements are needed, in this case, the new concept is *sensors*. All these sensors are connected in a *network* creating a system that can share information between devices to estimate the location. But why IPS?.

There is a need for locating in the interior area. Satellite positioning has not been functioning in closed areas, therefore it has led to the emergence of new technologies to make such a positioning possible. Indoor positioning is more complex than the outdoor positioning (by means

of GPS) as the implementation of technologies used indoors may require some additional infrastructure. IPS could be designed to search for items or groups of items in the supermarket, to find the appropriate office within the building of an institution, to orient oneself within the hospital complex, to find certain stores within the mall, to get around easily at the airport, museum, etc. And based on that, what technologies are available for developing IPS, what are their possibilities?

Figure 2.1: *Left:* IPS Market in USD by application. *Right:* IPS Market in USD by technology.

There have been several attempts to create a huge research field in this topic leading with numerous studies [10]. Most of the algorithms and techniques have been implemented for outdoor scenarios having as reference the Global Positioning System (GPS) and the new systems that will appear fully deployed in 2020 as Beidou (Asia and the West Pacific) or Galileo (European Union) and others systems that operates in Russia (GLONASS), India (IRNSS) and Japan (QZSS). However, how indoor environments are completely different and for now GPS and other regional systems don't provide a very accurate position estimation. Thus, several technologies have been proposed and implemented to improve positioning and navigation indoors. Some of them are Infrared (IR), Ultrasound, Audible Sound, Magnetic, Optical and Vision, Radio Frequency (RF), Visible Light, Pedestrian Dead Reckoning (PDR)/Inertial Navigation System (INS) and Hybrid. The RF technologies include Bluetooth, Ultra-wideband (UWB), Wireless Sensor Network (WSN), Wireless Local Area Network (WLAN), Radio-Frequency Identification (RFID) and Near Field Communication (NFC). In the case of Optical and Vision sensors several types of cameras can be used as Normal Field of View Cameras (with a different aperture angle), Fisheye Cameras (180° field of view in a distorted image), RGB-D cameras (depth and infrared information) and 360° cameras (that provide the 360° information in a projected image).

In addition, positioning techniques applied in indoor positioning systems include the signal properties and positioning algorithms. The prevalent signal properties are Angle of Arrival (AOA), Time of Arrival (TOA), Time Difference of Arrival (TDOA) and Received Signal Strength Indication (RSSI), while the positioning algorithms are Triangulation, Trilateration, Proximity and Scene Analysis/ Fingerprinting. In the other hand, Optical and Vision sensors work in a complete different way and exploit the use of Computer Vision, Machine/Deep Learning (ML and DL) and Artificial Intelligent (AI) techniques.

The range of potential applications is huge and researchers try to provide solutions looking a tradeoff between precision and cost. Some sensors provide very accurate measurements but the prize increase exponentially accordingly. In this way the computational cost, normally, is more reduced due to the developed algorithms are able to provide good estimations using the sensor measurements. Regarding low cost sensors, the measurements are not very accurate but using more complex processing algorithms the same accuracy as the provided by the expensive sensors can be achieved. Some applications can be framed in these topics:

- *Indoor Navigation:* In this topic, the main idea is provide solutions able to provide the real time location of persons or objects. User can interact with an interface to know where they are located and the system can provide information about the environment around the user.
- *Autonomous Robot navigation:* Robot navigation means the robot's ability to determine its own position in its frame of reference and then to plan a path towards some goal location. In order to navigate in its environment, the robot or any other mobility device requires representation, i.e. a map of the environment and the ability to interpret that representation. Avoiding dangerous situations such as collisions and unsafe conditions (temperature, radiation, exposure to weather, etc.) comes first, but if the robot has a purpose that relates to specific places in the robot environment, it must find those places.
- *Asset tracking:* Asset tracking refers to the method of tracking physical assets, either by scanning barcode labels attached to the assets or by using tags using GPS, BLE or RFID which broadcast their location. These technologies can also be used for indoor tracking of persons wearing a tag.
- *Proximity Marketing:* Proximity marketing can be described as the communication with customers at the right place, the right time and with highly relevant and personalized

messages, on their smartphones, be it greeting at the entry points, special offers in the store aisles, or getting feedback on a new product.

- *Warehouse/Workforce Monitoring*: A warehouse management system (WMS) is a software application, designed to support and optimize warehouse functionality and distribution center management. These systems facilitate management in their daily planning, organizing, staffing, directing, and controlling the utilization of available resources, to move and store materials into, within, and out of a warehouse, while supporting staff in the performance of material movement and storage in and around a warehouse.
- *Augmented/Virtual Reality*: Augmented reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real-world are "augmented" by computer-generated perceptual information, sometimes across multiple sensory modalities, including visual, auditory, haptic, somatosensory, and olfactory. The overlaid sensory information can be constructive (i.e. additive to the natural environment) or destructive (i.e. masking of the natural environment) and is seamlessly interwoven with the physical world such that it is perceived as an immersive aspect of the real environment. In this way, augmented reality alters one's ongoing perception of a real-world environment, whereas virtual reality completely replaces the user's real-world environment with a simulated one.
- *Smart Shopping*: Defined as a tendency for consumers to invest considerable time and effort in seeking and utilizing promotion-related information to achieve price savings. This concept uses the relationships to other related marketplace traits and behaviors as well as shopping-specific outcomes.
- *Behaviour analysis*: Behaviorism is a systematic approach to understanding the behavior of humans and other animals. It assumes that all behaviors are either reflexes produced by a response to certain stimuli in the environment, or a consequence of that individual's history, including especially reinforcement and punishment, together with the individual's current motivational state and controlling stimuli. In these applications, sensors try to infer a prediction about what the person will act in the next moment or what the person did using the collected information.

Lots of applications have been presented, but how each different sensor can be used to get a reliable estimation of the position of a person or an object in an indoor environment?. The next sections are intended to provide information about how each technology can be employed, what type of measurements are provided and what types of algorithms can be used to solve this question.

2.1 Technologies employed in IPS

Several technologies have been employed to perform indoor positioning with promising results. Noninvasive systems grew quickly in this topic due to its cost and availability in the market. In this context wireless technologies are the most used for tacking using different measured parameters from the received signal in a node. The main problem that appear using these approaches is the necessity of an active or passive device allocated in the body of the person tracked. This problem is very important in practice due to the availability of these systems in an open scenario is quite reduced depending on the technology employed.

In contrast, the other main type of technologies employed are vision-based systems. The combination of several cameras in different places can monitor an extensive area to apply positioning vision algorithms. In this case the way to work is completely different using as input data the image taken by the cameras and performing Computer Vision techniques. These techniques have been extensively studied and the growth in the computational power in computers have modified completely all these techniques in recent years. Machine/Deep Learning algorithms are showing the state-of-the-art results in image processing in task such as object detection, object tracking and object recognition.

2.1.1 Wireless Based Technologies used in IPS

RFID works using electromagnetic transmissions to an RF compatible receiver and is now being seen as a means of enhancing data handling processes. However, the application range is very limited due to type of signals employed. Normally, the reading range is around 1–2 meters and the cost of the receivers is very high. The typical configuration consists of a RFID reader that allows the connection of several multiple antennas. Data collected is sent through the network a collected by a final working station that contains all the algorithms for IPS purposes. Tags should be allocated in moving objects of interest to be detected by the antennas inside deployed scenario). An example is shown in Figure 2.2.

Another common technology employed in IPS is Wireless Local Area Network (WLAN), which operates in the 2.4 GHz Industrial, Scientific and Medical (ISM) band. Main characteristics in order to use this technology is that the increasing of bit rate and the high coverage range (50-100 meters) this protocol (IEEE802.11) is the actual dominant local wireless network standard. Then, it is possible use the node infrastructure of this network for indoor positioning

and tracking having a final server collecting all the measurements received by several nodes. It is very common use this technology with the Received Signal Strength Indication (RSSI) or Channel State Information (CSI) parameters having an accuracy between 3-30 meters. The combination with more complex algorithms can reduce this value to under 1 meter in some concrete situations and configurations. Typical network architecture is drawn in Figure 2.3.

Figure 2.2: Typical RFID architecture. Several antennas are deployed connected typically to one RFID reader that is able to work with more than tens of them. RFID tags are allocated in moving objects or persons and work passively through the monitored environment.

Bluetooth is another wireless technology that need the mention in this field. It works in the 2.4 GHz ISM band as well. Main differences exist between this technology and WLANs. The bit rate is slower (1-3 Mbps) and the range is typically in the range of 10-15 meters. Main characteristics are the cost, ubiquity (embedded in several devices like phones, tablets...) and is able to support several networking services, more than only IP. The Bluetooth tags are normally of a small size and have unique IDs that can be used for indoor localization.

Bluetooth technology is mainly used in small-scale positioning, such as single-room or warehouse. Network architectures that employs Bluetooth technologies are similar to WLAN architectures replacing WLAN access points with Bluetooth access points.

ZigBee solutions are widely applied in lots of areas, researches and applications. This low-cost technology is able to transmit data with low-power and a reduced data rate inside a wireless mesh that interact between nodes and data collectors. The standard come from the IEEE802.15.4-2003 and was improved with the new standard IEEE802.14.4-2006 to insert more capabilities and features. This standard is normally employed like in WLAN and Blue-

tooth solutions using some nodes to collect data in different locations and a personal device with this technology to send the important data.

Figure 2.3: WLAN indoor positioning architecture. Access points are deployed covering big indoor areas and are connected to the same network to share information between them and a global server with all the positioning methods.

Wireless Sensor Networks (WSN) are the main solution using the previous technologies (Figure 2.4). Normally, one access point in a Wireless Sensor Network is configured to receive data using one or more technologies at the same time performing some synchronization algorithm between the received signals. The availability of several mini-embedded computer solutions such as Raspberry-Pi, Arduino among others have extensively increase the creation of simple/complex networks at a low cost improving the performance of Indoor Positioning Solutions. Some of these System on Chip (SoC) solutions include more than one wireless technology on board directly without the inclusion of more sensors independently.

A different technology extensively used is Ultra-Wide Band (UWB) which have some important differences in contrast to the presented WLAN, Bluetooth and Zigbee technologies. In this case, a very narrow pulses of nanoseconds are sent to transmit the data and has the magnitude of the bandwidth of the signal (Figure 2.5). This technology can be used also for accurate indoor positioning and these systems has some advantages such low power consumption, avoidance of multi-path effects, penetrating power (due to the bandwidth employed), low complexity and high security.

Figure 2.4: Wireless Sensor Networks advantages.

A not very extended approach but also useful for these purposes is the Global System Mobile (GSM) with Code Division Multiple Access (CDMA) in cellular mobile networks. This approach is most extended for outdoor scenarios but can be applied also in indoor environment if accurate positioning is not a very strong requirement. Using Cell-ID method or Enhanced Observed Time Difference (E-OTD) provide an accuracy in the range of 50-200 meters in outdoor. Regarding indoor environments if cells are high size (entire rooms) RSSI can be used with several base stations along the monitored area or building using a phone ported by the user. The complexity of this type of signals and the cost of the base stations are neglecting the use of these solutions.

In addition, ultrasonic technologies by ranging can be used in combination with triangulation to determine the position of the user in indoor scenarios. These systems present a very simple configuration and network structure but are highly affected by multi-path effects increasing the cost of the devices to get more accurate results.

Figure 2.5: UWB spectrum compared with narrow-band and spread-spectrum technologies.

Finally, infrared is another technology employed in IPS systems. Infrared modulated IR emit a ray that is collected by an optical sensor installed in the node collector. The architecture is very simple and is low cost presenting in addition high estimation accuracy due to the necessity of line of sight between the user device and the receiver. This is also in contrast the main problem because if several obstacles are in the middle of the signal no measurements are collected and the devices only provide short distance ranges (Figure 2.6).

Figure 2.6: IPS Infrared Architecture using LEDs information. A device needs a sensor able to collect infrared information sent by the main architecture. The receiver processes the information to provide the actual position.

2.1.2 Inertial Navigation Systems (INS) and Pedestrian Dead Reckoning (PDR)

An Inertial Navigation System (INS) is comprised of an Inertial Measurement Unit (IMU) and a System on Chip (SoC) or computer able to collect, control and process measurements from the sensors. IMUs typically contain three orthogonal rate gyroscopes and three orthogonal accelerometers and in some cases, magnetometers are also included (Figure 2.7). Gyroscopes measurements of inertial angular rate are used to determine the orientation of a triad of accelerometers. The accelerometer measurements, in turn, are integrated to estimate vehicle velocity and position. Magnetometers can measure the magnetic field and this information can be used to estimate the absolute orientation with respect to the magnetic north of the earth.

Figure 2.7: Typical Inertial Navigation System. Accelerometer and Gyroscope measurements can be combined to estimate a lot of other important parameter to get position and orientation of a moving object. Initial conditions are necessary to start the system with the correct parameters.

By processing signals from these devices, the computer calculates the device position, orientation, and velocity with reference to its starting point. To initialize the INS an operator, a Global Navigation Satellite System (GNSS), or some other external input provides an INS with its initial position and velocity. Subsequently an INS can compute the position and velocity of a moving object by integrating information received from its IMU. Others sensors such as Barometers and Temperature sensors are also included to estimate attitude and temperature respectively that can affect other measurements in the system.

The advantage of an INS is that after initialization it requires no external references to determine its position, orientation, or velocity. So, it is a self-contained, immune to jamming,

inherently stealthy all weather positioning and attitude system. Moreover, the most important advantage of INS is that velocity, position and attitude of the vehicle can be provided with abundant dynamic information and excellent short-term performance. The main disadvantages are the requirements in the system.

If high accuracy in estimations is needed the price of the sensors increase exponentially but the algorithmic solution should be simpler and stable. In the other hand, there exists very cheaper sensors that contains accelerometer, gyroscope and magnetometer in the same chip presenting a very reduced size and can cost around 15\$. In the second case the algorithmic problem is the most difficult part due to several noise sources can affect the raw measurements.

A different approach that also use IMUs is named pedestrian inertial navigation systems that are normally based on pedestrian dead reckoning (PDR) algorithm. In these systems acceleration is not treated integrating the values to get the velocity. Based on the human behaviors and characteristics using a combination of sensors (accelerometer, gyroscope, magnetometer, barometer, among others) some useful information is extracted regarding common patterns from the sensor measurements like waveform, repetitions. All these features associated to the walking movement can be used to estimate the velocity, orientation and step length between many other parameters related to movement. Moreover, regarding orientation, the combination of several sensors is the common approach employed but the current personal device position affects severely the performance of these algorithms. The most challenging part is to estimate accurately a common reference system independent of the position of the device in the body to obtain an accurate absolute orientation reference system. A typical architecture for PDR IPS is represented in Figure 2.8.

Figure 2.8: Typical Pedestrian Dead Reckoning Architecture. All the information is extracted from sensors and a sensor fusion and a step detection algorithm are combining to get all the necessary parameters to know person position and orientation.

2.1.3 Image Based Technologies used in IPS

Massive amount of security cameras, along with other sensors, have been deployed to monitor critical infrastructures such as: military bases, airports, power plants, banking, campuses, etc. Manual monitoring by human operator is an inefficient solution or even unpractical because human resource is expensive and has limited ability. Intelligent surveillance system (ISS) is envisioned to automatically monitor the environment or infrastructures with less or without human intervention. Such monitoring tasks include automatically detecting and tracking object (like human or vehicle) and performing further analysis and actions. Signal processing, image processing, and artificial intelligence (machine/deep learning) techniques play an important role to develop such intelligent system [11].

ISS is an interdisciplinary topic that involves electronic (sensing device), computer vision and pattern recognition, artificial intelligence (machine/deep learning), networking, communication and other areas. Intelligent surveillance system is promising to be implemented in various environments and applications. Some typical applications are listed as follows.

- Home security and intrusion detection.
- Home care and safety.
- Public transport area such as airport, seaport, bus/train terminal.
- Public area such as colleges, campuses, governmental building.
- Traffic monitoring.
- Crowd management and analysis.
- Pedestrian detection and autonomous car.
- Remote military surveillance, border monitoring, perimeter surveillance for power plant, company, etc.

Intelligent surveillance systems may play a significant role in security and safety in public, as well as in private domain. However, it is highly challenging due to some practical issues, such as:

- Performance, such as the system accuracy
- Robustness again real-world issues such as illumination variation, clutter, occlusion, weather change, camouflage, etc.
- Reliability
- Real time constrain, the system should fast enough
- Cost effective

Due to the high amount of problems related to image capture several types of cameras that used different sensors are able in the market in order to measure important information that is better in concrete scenarios. The optical lens of the camera plays an important part in this field due to there are cameras able to collect normal field o view (NFoV) images or other distorted projections using more complex optical sensors able to collect information around 180° or the new 360° cameras able to get information around all the environment angles in the space. In the other hand, some cameras are provided with infrared or other lighting sensors able to collect depth information or infrared and temperature images.

Visible camera such as Closed Circuit Television (CCTV) are the most common devices for surveillance systems (Figure 2.9). CCTV is the use of video cameras to transmit a signal to a specific place, on a limited set of monitors. It differs from broadcast television in that the signal is not openly transmitted, though it may employ point to point (P2P), point to multipoint (P2MP), or mesh wired or wireless links.

Figure 2.9: Left and center: Examples of CCTV cameras used in surveillance systems. Right: Typical CCTV cameras installation for surveillance and traffic control.

It has long been in use to monitor environments, people, events and activities. Extensive studies have been conducted to automatically analyze data (image or video) from surveillance camera. Much of these studies have been discussed in several review papers with main drawbacks background-foreground segmentation, objects detection and classification, tracking, and behavioral analysis.

Actually, other type of cameras are gaining an important interest due to its availability to get other different image projections or important visual information such as depth. Normally one CCTV camera has a FoV between 70° and 90° . Other cameras have different lens that can extend this range until 135° but reducing the distant capable from the camera to see far objects (Figure 2.10).

Figure 2.10: Camera focal length depending on aperture angle (FoV) in CCTV cameras.

A fisheye lens is an ultra-wide-angle lens that produces strong visual distortion intended to create a wide panoramic or hemispherical image (Figure 2.11). Fisheye lenses achieve extremely wide angles of view. Instead of producing images with straight lines of perspective (rectilinear images), fisheye lenses use a special mapping which gives images a characteristic convex non-rectilinear appearance. These cameras can provide a 180° image containing information about an extensive area. These cameras are normally located at the ceiling or in the wall with the lens perpendicular to the floor.

Different sensor modalities other than visible cameras have been explored also for surveillance systems such as infrared and thermal cameras, radar (radio detection and ranging), lidar (light

detection and ranging), etc. Other review papers have discussed also different techniques for sensor fusion to improve the system performance.

Figure 2.11: Example of Fisheye images. Top left: Hemispheric image from a ceiling camera. Top right: N FoV projection of 65° over a part of the image. Bottom: Panoramic projection from the Hemispheric image (loss of information in the center of the image)

Visible camera is working well only in the environment that has enough illumination or light intensity, for example during in daytime. In the environment with low light intensity or during the night, visible camera cannot capture the scene effectively. In this case, there are two possible solutions: using infrared camera or thermal camera. Object (like human) that has contrast temperature with the surrounding environment is much easier to distinguish in the thermal or infrared camera image than in the visible camera image. Both cameras capture infrared radiation that is invisible for human eye.

Range sensing is an interesting sensor modality due to its accuracy, large field of view and robustness with respect to illumination changes. Such range sensing includes radar (radio detection and ranging) and lidar (light detection and ranging). Radar uses radio waves for sensing, while lidar uses light or laser. In range data, changes in the background can be easily filtered out by excluding all data outside of the tracking area. One drawback is that range

data is generally less informative than vision data for person or object recognition.

3D information can be collected using 3D/Depth-Cameras that provide directly 3d data, Stereo-Cameras that consists (in most cases) of two cameras separated a concrete distance that can approximate a depth map using both collected images and finally RGBD-Cameras which contains a sensor that gives depth and color information. In most cases, this refers to a Kinect-Style camera [12] (or Primesense [13], Realsense [14]), but a stereo-camera could also create colored point clouds (if one camera is a RGB camera or if you have an additional cameras).

There is also a distinction of passive setups (like a standard stereo camera) or active cameras with some kind of projectors. Active cameras can use a speckle projector to create a texture that is then captured by a camera (Kinect v1, Primesense, Asus Xtion Pro) or use flash and Time-based computations (Time of Flight) to measure the distance (Kinect v2, Picoflex [15]).

Figure 2.12: Left: Kinect v1 camera (infrared speckle pattern). Center: MYNT Eye 3D (stereo depth camera sensor) [1]. Right: Kinect v2 camera (Time of Flight measurement.)

For example, the PrimeSense sensor projects an infrared speckle pattern. The projected pattern is then captured by an infrared camera in the sensor, and compared part-by-part to reference patterns stored in the device. These patterns were captured previously at known depths. The sensor then estimates the per-pixel depth based on which reference patterns the projected pattern matches best.

The depth data provided by the infrared sensor is then correlated to a calibrated RGB camera. This yields an RGB image with a depth associated with each pixel. A popular unified representation of this data is a point cloud that is a collection of points in three-dimensional space, where each point can have additional features associated with it. With an RGB-D sensor, the color can be one such feature. Additionally, approximated surface normal's are also often stored with each point in a point cloud.

Figure 2.13: Example of quotidian actions recorded with Kinect v2 camera. Top part represents the RGB image collected by the camera. The bottom part contains the Depth image extracted with the camera and the estimated skeletons of the persons (Kinect v2 camera intrinsic feature) presented in the image.

Finally the last type of camera important to present in this work is the omnidirectional camera, also known as 360° camera. It is a camera having a field of view that covers approximately the entire sphere or at least a full circle in the horizontal plane. Omnidirectional cameras are important in areas where large visual field coverage is needed, such as in panoramic photography and robotics.

A camera normally has a field of view that ranges from a few degrees to, at most, 180°. This means that it captures, at most, light falling onto the camera focal point through a hemisphere. In contrast, an ideal omnidirectional camera captures light from all directions falling onto the focal point, covering a full sphere. In practice, however, most omnidirectional cameras cover only almost the full sphere and many cameras which are referred to as omnidirectional cover only approximately a hemisphere, or the full 360° along the equator of the sphere but excluding the top and bottom of the sphere. In the case that they cover the full sphere, the captured light rays do not intersect exactly in a single focal point. Figure 2.14 present different representations of the 360° information.

Figure 2.14: Most common 360° camera projections. From left to right: Two Fisheye 180° projections, Equirectangular projection (rectangular image that contains all the 360° information), Domemaster (representation in a 3D sphere) and CubeMap (6 NFOV projections in a cube).

Various techniques can be used to generate 360-degree images like cameras with one lens (these models are used with a fisheye lens), cameras with two lenses (the most common type, they can accommodate a full 360-degree angle with two lenses facing each other, problems are often caused by stitching errors), cameras with more than two lenses (the more lenses are installed in the camera, the more difficult it becomes for the software to combine the individual images, very expensive solution), camera rigs (camera rigs are mostly used for the attachment of 6 conventional Actioncams, for example GoPro) and mosaic based cameras (several "normal" cameras are combined in a network and the individual images are then joined together like mosaic stones to form an omnidirectional overall image). Figure 2.15 contains some different camera types mentioned in this paragraph.

Figure 2.15: Different 360° camera types. From left to right: Rig camera (combination of 6 cameras) [2], mosaic system (one normal camera rotates along several angles stitching images), two fisheye 180° lens joined together with a stitching post-processing [3] and camera with more than two spherical lenses [4].

Intuitively, combining multiple sensors will provide more accurate information about the targeted object. Multiple sensors might be homogeneous (same modality, such as multiple cameras) or heterogeneous (different modalities). Some sensor modalities are intuitively closed and complementary, such as visible, infrared and thermal camera, since they capture information in 2D image perspective. Similarly, both lidar and radar capture information in range domain (2D or 3D). Sensor fusion may happen at low-level (data fusion), high level (decision fusion) or in between.

In data fusion, each sensor sends its original measurement to the fusion center, and then the center makes the decision about the event. In decision fusion, each sensor makes its own decision based on its own measurement, and then the fusion center makes the final decision based on all individual decisions (for example using majority voting). Each fusion scenario has its own advantages and drawbacks. Challenges in sensor fusion are how to handle different data modalities (visual, audio, radio signal...), data imperfection, conflicting data, sensor topology, etc.

Chapter 3

Machine and Deep Learning

Learning algorithms are actually one of the key points in data analysis in order to extract hidden knowledge from huge amounts of data. Depending on the nature of the problem some algorithms work better than others. In traditional Machine Learning based projects the problem is normally divided into two principal steps, features engineering and learning phase. The initial phase consists on getting from the data the important values performed some cleaning process, data manipulation and data adaptation to get rich features that can be used to train a learning algorithm efficiently. A lot of research exists about this topic due to the numerous types of data available in the world. The second step is the algorithm selection, training and validation. In this part the algorithm is fine tuned to get the best performance before deploying it in the real working scenario.

In the other hand Deep Learning algorithms are now breaking the state-of-the-art results in several tasks like detection, recognition, segmentation and classification. Deep neural networks are able to learn hidden patterns in a non-linear manner providing a very powerful tool to learn efficiently from complex data inputs. Deep neural networks can be applied to all the types of data (audio, image, raw data vectors...) and are able (depending on the network used) to extract automatically the feature vectors avoiding this manual step in the process.

In this section a complete review of the main important machine learning and deep learning networks is provided. In addition, a short description of the algorithms and the main applications are described. Metrics and parameters to evaluate the algorithms are also provided.

3.1 Machine learning

This concept is composed by a set of algorithms for data analysis that automatize the analytic building a model using the data under study. Using algorithms that iteratively acts over the data, Machine/Deep Learning allows computers learn hidden knowledge without looking it manually. But what is Machine Learning?

Figure 3.1 collect some definition about what is Machine Learning. It is clear from all the different definitions that "data" is one of the main concepts involved in Machine Learning. The second important concept is "to learn" or "act" over the data. With these definitions now is more clear what is Machine Learning, but, how this algorithms works?. Who to start with them?. Which is the process to follow?. All these algorithms present the same project structure divided in several steps. These steps are presented in Figure 3.2.

Formal definition

"Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions."

"Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world." – Nvidia

"Machine learning is the science of getting computers to act without being explicitly programmed." – Stanford

"Machine learning is based on algorithms that can learn from data without relying on rules-based programming."- McKinsey & Co.

"Machine learning algorithms can figure out how to perform important tasks by generalizing from examples." – University of Washington

"The field of Machine Learning seeks to answer the question "How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?" – Carnegie Mellon University

Figure 3.1: Definitions of what is machine learning by different known companies and universities.

Data acquisition is the task where the data that will be used to train our algorithms is collected or obtained from existing datasets. Data cleaning is related to the task where the data is inspected in order to assure that all the information is correct or check if is necessary remove or perform modifications if exists a lack of values or the data is not perfectly adjusted to the problem under study. Model training and testing are the operations performed using a learning algorithm to learn over the data and test if the model learned has the accuracy necessary to solve the problem. Model deployment refers to the use of the trained model in the final application to predict new values or information.

Figure 3.2: Machine Learning project chain starting from data acquisition and cleaning, passing through the learning phase and finalizing with the model deployment.

There are many different types of machine learning algorithms, with hundreds published each day, and they are typically grouped by either learning style (supervised, unsupervised, reinforcement) or by similarity in form or function (i.e. classification, regression, clustering, deep learning, etc.). Regardless of learning style (Figure 3.3) or function, all combinations of machine learning algorithms consist of the following:

- Representation (a set of classifiers or the language that a computer understands)
- Evaluation (objective/scoring function)
- Optimization (search method)

Figure 3.3: Machine Learning techniques grouped by style and functions.

Supervised Learning is used when a labeled dataset is available. Labeled refer to a set of data (multimedia) each one with the corresponded tag/description. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output. The inputs are a series of data vectors (features) and the output are the corresponded label estimated from the learning over the data.

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data. These problems can be solved as a classification (output is a category) or a regression (output is a real value).

For Unsupervised Learning the problems to solve include a dataset composed by a set of data but all this data is not labeled, it is not known the correct value. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more

about the data. Algorithms try to look for common patterns in the data to associate it (you want to discover rules that describe large portions of your data) or clustering (you want to discover the inherent groupings in the data).

For Reinforce Learning no dataset is available but it is known previously the behavior and rules involved in the problem. These algorithms interact with the environment inputs (states) and try figure what is the best action to do. Once an action is decided a feedback (reward/penalty) is received.

Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y).

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Dimensionality Reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables.

Most machine learning challenges relate to handling your data and finding the right model. Data comes in all shapes and sizes. Real-world datasets can be messy, incomplete, and in a variety of formats. You might just have simple numeric data. But sometimes you're combining several different data types, such as sensor signals, text, and streaming images from a camera. Preprocessing your data might require specialized knowledge and tools. For example, to select features to train an object detection algorithm requires specialized knowledge of image processing. Different types of data require different approaches to preprocessing.

It takes time to find the best model to fit the data. Choosing the right model is a balancing act. Highly flexible models tend to overfit data by modeling minor variations that could be noise. On the other hand, simple models may assume too much. There are always trade-offs between model speed, accuracy, and complexity. Selecting a machine learning algorithm is a process of trial and error. It's also a trade-off between specific characteristics of the algorithms, such as speed of training, memory usage, predictive accuracy on new data and transparency or interpretability (how easily you can understand the reasons an algorithm

makes its predictions).

Lots of algorithms have been developed to learn from data and have different ideas, parameters and the way to solve the problem is different. But, all of them have something in common, it is needed perform a parameter selection to fit the algorithm to the concrete problem. Depending on the problem to solve some of these algorithms are:

- Supervised Classification: Logistic Regression, k Nearest Neighbor (kNN), Support Vector Machine (SVM), Naïve Bayes, Decision Tree, Bagged and Boosted Decision Trees, Discriminant Analysis, Simple Neural Network.
- Regression: Linear Regression, Nonlinear Regression, Gaussian Process Regression Model, SVM Regression, Generalized Linear Model, Regression Tree.
- Unsupervised Clustering: k-Means, k-Medoids, Hierarchical Clustering, Self-Organizing Map, Fuzzy c-Means, Gaussian Mixture Model.
- Reinforce Learning: Policy optimization or policy-iteration methods, Policy Gradient, Asynchronous Advantage Actor-Critic (A3C), Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), Q-learning.

Using these algorithms is necessary get some useful metric able to represented how the algorithm have learn over the selected data to assure a good performance before the final algorithm deployment. Classification problems are perhaps the most common type of Machine Learning problem and as such there are a myriad of metrics that can be used to evaluate predictions for these problems (Figure 3.4). The following metrics are the most used:

- Classification accuracy is the number of correct predictions made as a ratio of all predictions made.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.1)$$

Where TP is True Positives, FP False Positives, TN True Negatives and FN False Negatives.

- Logarithmic Loss works by penalizing the false classifications. When working with Log Loss, the classifier must assign probability to each class for all the samples.

$$LogLoss = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij}) \quad (3.2)$$

Where N is the number of samples, M is the number of possible labels, y_{ij} takes value 1 if label j is the correct classification for instance i , and 0 otherwise and p_{ij} is the model probability of assigning label j to instance i .

- Confusion Matrix gives us a matrix as output and describes the complete performance of the model comparing all the data involved and presenting the correct and incorrect number of asserts and failures between classes.
- Area Under Curve (AUC) is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example.
- F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is $[0, 1]$. It tells you how precise your classifier is, as well as how robust it is.

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (3.3)$$

But, for regression problems a different type of metrics are used due to its necessary measure some distance or difference between predicted and real values involved in the learning process.

- Mean Absolute Error (MAE) is the sum of the absolute differences between predictions and actual values. It gives an idea of how wrong the predictions were. The measure gives an idea of the magnitude of the error, but no idea of the direction.

$$MAE = \frac{1}{n} \sum_{t=1}^n |e_t| \quad (3.4)$$

- Mean Squared Error (MSE) takes the average of the square of the difference between the original values and the predicted values. As, we take square of the error, the effect of larger errors become more pronounced then smaller error.

$$MSE = \frac{1}{n} \sum_{t=1}^n e_t^2 \quad (3.5)$$

- R^2 (or R Squared) metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination. This is a value between 0 and 1 for no-fit and perfect fit respectively.

Figure 3.4: Common Metrics for different types of Machine Learning problems.

Finally, some other important tips related to Machine Learning are model improvement and feature selection. Improving a model means increasing its accuracy and predictive power and preventing overfitting (when the model cannot distinguish between data and noise). Model improvement involves feature engineering (feature selection and transformation) and hyperparameters tuning. The process is drawn in Figure 3.5.

Feature selection is the task of identify the most relevant features, or variables, that provide the best predictive power in modeling your data. This could mean adding variables to the model or removing variables that do not improve model performance. By feature transformation the existing features are transformed into new features using techniques such as principal component analysis, non-negative matrix factorization, and factor analysis. Finally, hyperparameters tuning is the process of identifying the set of parameters that provides the best model. Hyperparameters control how a machine learning algorithm fits the model to the data.

Feature selection is one of the most important tasks in machine learning. It's especially useful when you're dealing with high dimensional data or when your dataset contains a large number of features and a limited number of observations. Reducing features also saves storage and computation time and makes your results easier to understand.

Figure 3.5: Scheme for Feature Selection, Transformation and Hyperparameters Tuning in a Machine Learning problem.

3.2 Deep Learning

Deep learning is a type of machine learning in which a model learns to perform classification, regression or other tasks directly from images, text, or sounds between among others. Deep learning is usually implemented using a neural network architecture. The term "deep" refers to the number of layers in the network, the more layers, the deeper the network. Traditional neural networks contain only 2 or 3 layers, while deep networks can have hundreds.

What Makes Deep Learning state-of-the-art?. In a word, accuracy. Advanced tools and techniques have dramatically improved deep learning algorithms to the point where they can outperform humans at classifying images, win against the world's best GO player, or enable a voice-controlled assistant like Amazon Echo and Google Home to maintain a conversation.

Deep learning approaches can be categorized like in machine learning and can be seen as an extension of them but deeper. Supervised, semi-supervised or partially supervised, and unsupervised. In addition, there is another category of learning approach called Reinforcement Learning (RL) or Deep RL (DRL) which are often discussed under the scope of semi-supervised or sometimes under unsupervised learning approaches.

Figure 3.6: Top-5 error on ImageNet Dataset (bigger world dataset of images). The accuracy in 2015 using Deep Learning networks equal or improve the accuracy presented by human at classification level.

Supervised learning is a learning technique that uses labeled data. In the case of supervised DL approaches, the environment has a set of inputs and corresponding outputs. For example, if for input x_t , the intelligent agent predicts $\hat{y}_t = f(x_t)$, the agent will receive a loss value $l(y_t, \hat{y}_t)$. The agent will then iteratively modify the network parameters for a better approximation of the desired outputs. After successful training, the agent will be able to get the correct answers to questions from the environment. There are different supervised learning approaches for deep learning, including Deep Neural Networks (DNN), Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN).

Semi-supervised learning is learning that occurs based on partially labeled datasets. In some cases, Deep Reinforce Learning (DRL) and Generative Adversarial Networks (GAN) are used as semi-supervised learning techniques. Additionally, RNN, are used for semi-supervised learning as well.

Unsupervised learning systems are ones that can work without the presence of data labels. In this case, the agent learns the internal representation or important features to discover unknown relationships or structure within the input data. Often clustering, dimensionality reduction, and generative techniques are considered as unsupervised learning approaches. There are several members of the deep learning family that are good at clustering and non-linear dimensionality reduction, including Auto-Encoders (AE), Restricted Boltzmann Ma-

chines (RBM) and GANs. In addition, RNNs and RL, are also used for unsupervised learning in many application domains.

Deep Reinforcement Learning is a learning technique for use in unknown environments. In RL, we do not have a straight forward loss function, thus making learning harder compared to traditional supervised approaches. The fundamental differences between RL and supervised learning are, firstly, you do not have full access to the function you are trying to optimize, you must query them through interaction. Secondly, you are interacting with a state-based environment, the input depends on previous actions.

A key difference between traditional ML and DL is in how features are extracted. Traditional ML approaches use handcrafted engineering features by applying several feature extraction algorithms, and then apply the learning algorithms. On the other hand, in the case of DL, the features are learned automatically and are represented hierarchically in multiple levels. This is the strong point of DL against traditional machine learning approaches (Figure 3.7).

Figure 3.7: Machine Learning manual feature extraction against Deep Learning automatic feature extraction working graph.

At present, DL is being applied in almost all areas. As a result, this approach is often called a universal learning approach. Then after all these statements, why Deep Learning?

- **Universal Learning Approach:** The DL approach is sometimes called universal learning because it can be applied to almost any application domain.

- **Robust.** Deep learning approaches do not require the precisely designed feature. Instead, optimal features are automatically learned for the task at hand. As a result, the robustness to natural variations of the input data is achieved.
- **Generalization:** The same DL approach can be used in different applications or with different data types. This approach is often called transfer learning. In addition, this approach is helpful where the problem does not have sufficient available data.
- **Scalability:** The DL approach is highly scalable. Microsoft invented a deep network known as ResNet. This network contains 1202 layers and is often implemented at a supercomputing scale. There is a big initiative in the entire world to develop frameworks for networks like this, which can implement thousands of nodes.

Deep learning often requires hundreds of thousands or millions of data (features, images, audio, video...) for the best results. It's also computationally intensive and requires a high-performance GPU. Training a deep learning model can take hours, days, or weeks, depending on the size of the data and the amount of processing power you have available. Selecting a computational resource is a critical consideration when you set up your workflow.

Three technology enablers make this degree of accuracy possible in Deep Learning field. The first one is the easy access to massive sets of labeled data. Datasets such as ImageNet, COCO and PASCAL VoC are freely available, and are useful for training on many different types of objects.

The second is the increasing in computing power. High-performance GPUs accelerate the training of the massive amounts of data needed for deep learning, reducing training time from weeks to hours.

Finally, pretrained models built by experts freely accessible. Models such as AlexNet, VGGNet and ResNet can be retrained to perform new recognition tasks using a technique called transfer learning. While AlexNet was trained on 1.3 million high-resolution images to recognize 1000 different objects, accurate transfer learning can be achieved with much smaller datasets. ResNet is actually one of the most used network as a backbone to extract features from images.

A *Deep Neural Network (DNN)* combines multiple nonlinear processing layers, using simple elements operating in parallel and inspired by biological nervous systems. It consists of an input layer, several hidden layers, and an output layer. The layers are interconnected via

nodes, or neurons, with each hidden layer using the output of the previous layer as its input (Figure 3.8).

Figure 3.8: General architecture of a Fully Connected Deep Neural Network.

Let's say we have a set of images where each image contains one of four different categories of object, and we want the deep learning network to automatically recognize which object is in each image. We label the images in order to have training data for the network.

Using this training data, the network can then start to understand the object's specific features and associate them with the corresponding category. Each layer in the network takes in data from the previous layer, transforms it, and passes it on. The network increases the complexity and detail of what it is learning from layer to layer. Notice that the network learns directly from the data, we have no influence over what features are being learned.

A *Convolutional Neural Network (CNN, or ConvNet)* is one of the most popular algorithms for deep learning with images and video. Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between (Figure 3.9).

One important and different part in these networks are the Feature Detection Layers. These layers perform one of three types of operations on the data, convolution, pooling, or activation. Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images. Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn about. Activation using Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. These three operations are

repeated over tens or hundreds of layers, with each layer learning to detect different features.

Figure 3.9: General architecture of a CNN network for classification tasks.

After feature detection, the architecture of a CNN shifts to classification/regression. The next-to-last layer is a fully connected layer (FC) that outputs a vector of K dimensions where K is the number of classes that the network will be able to predict or the number of values that a regression will predict (for example, founding boxes coordinates). This vector contains the probabilities for each class of any image being classified or the final regress values. The final layer of the CNN architecture uses a Softmax function to provide the classification output or a Sigmoid among others to fit values in a certain range.

The idea behind *Recurrent Neural Networks (RNN)* is to make use of sequential information. In a traditional neural network, it is assumed that all inputs (and outputs) are independent of each other. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another important aspect of these type of networks is that can work with inputs of different lengths where DNNs and CNNs fix its input vector to a certain size outputting a concrete output size.

Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. The general scheme for a RNN is presented in Figure 3.10

Figure 3.10: General architecture of a RNN network. Sequential information is given as input and is feed into the network in different timesteps.

An *Autoencoder (AE)* is a deep neural network approach used for unsupervised feature learning with efficient data encoding and decoding. The main objective of autoencoder is to learn and represent (encoding) of the input data, typically for data dimensionality reduction, compression, fusion and many more. This autoencoder technique consists of two parts: The encoder and the decoder. In the encoding phase, the input samples are mapped usually in the lower dimensional features space with a constructive feature representation. This approach can be repeated until the desired feature dimensional space is reached. Whereas in the decoding phase, we regenerate actual features from lower dimensional features with reverse processing. The conceptual diagram of auto-encoder with encoding and decoding phases is shown in Figure 3.11.

Figure 3.11: General architecture of an Autoencoder where encoder/decoder parts are presented. In this example the autoencoder try to reproduce the input at the output having in the middle a compress representation of the data.

Usually, the feature space of the encoded vector has lower dimensions than the input feature space, which can be considered as the compressed representation of the input sample. A deep Auto encoder is constructed by extending the encoder and decoder with multiple hidden layers. Different advanced AE models are Variational Autoencoders (VAE) and Split-Brain Autoencoder (SB-AE). It is possible apply AE for unsupervised feature extraction and then apply other techniques for clustering those samples for generating labels. AE has been used as an encoding and decoding technique with or for other deep learning approaches, including CNN, DNN, RNN, and RL in the last decade.

Recently, deep learning-based generative models have become popular and shown enormous success in different application domains. Deep learning is a data-driven technique that performs better as the number of input samples increased. Due to this reason, learning with reusable feature representations from a huge number of the unlabeled datasets has become an active research area. This problem has been attempted to be solved by generating similar samples with a generative model. *Generative Adversarial Network (GAN)* is a deep learning approach recently invented by Goodfellow in 2014. GANs offer an alternative approach to maximum likelihood estimation techniques.

GAN is an unsupervised deep learning approach where two neural networks compete against each other in a zero-sum game. In the case of the image generation problem, the generator starts with Gaussian noise to generate images and the discriminator determines how good the generated images are. This process continues until the outputs of the generator become close to actual input samples. According to Figure 3.12, it can be considered that Discriminator (D) and Generator (G) two players playing the min-max game with a loss function.

Figure 3.12: General architecture of a Generative Adversarial Network (GAN) with the Generator (G) and the Discriminator (D) that play together in the Loss function of a GAN.

This learning algorithm has been applied in the different domain of applications. GANs are mostly used for generating a photo-realistic images, semantic segmentation, generating or synthesizing images from a text description, etc... Recently, GANs are used for speech analysis and enhancement, symbolic-domain and music generation which performs comparably against solutions with RNNs. Some other applications of GAN include pose estimation, photo editing network, anomaly detection, unsupervised image-to-image translation or the actual famous Deep Fakes.

Finally, *Deep Reinforce Learning (DRL)* is a learning approach which learns to act with general sense from the unknown real environment. The conceptual diagram for DRL approach is shown in Figure 3.13. RL can be applied in a different scope of field, including fundamental Sciences for decision making, Machine learning from a computer science point of view, in the field of engineering and mathematics, optimal control, robotics control, power station control, wind turbines, and Neuroscience the reward strategy is widely studied in the last couple of decades. It is also applied in economic utility or game theory for making better decisions and for investment choices. The psychological concept of classical conditioning is how animals learn. Reinforcement learning is a technique for what to do and how to match a situation to an action. Reinforcement learning is different from supervised learning technique and other kinds of learning approaches studies recently, including traditional machine learning, statistical pattern recognition, and ANN.

Figure 3.13: Deep Reinforce Learning conceptual system diagram architecture.

Chapter 4

Overview of the complete proposed system architecture

Depending on the technology used (Wireless, IMUs or visual) the selected parameters to use are completely different as well as the algorithms employed to perform positioning and tracking in indoor environments. Regarding wireless signals Received Signal Strength Indicator (RSSI) have been used due to its availability in all the employed technologies (Bluetooth, WiFi and XBee) which provide information about the received signal power in the receiver. Another interesting parameter to use is the Channel State Indicator (CSI) that is information to describe how a signal propagates from the transmitter to the receiver and represents the combined effect of, for example, scattering, fading, and power decay with distance but some of the devices employed lack on provide this parameter.

Others technologies use the same concept of RSSI but in different ways such as in UWB, IR and Ultrasound systems. Based on the measurements collected the position can be inferred in the same ways as in WLAN, Bluetooth, ZigBee technologies. The problem of UWB technology is the short range where can be used avoiding the use in bigger areas without the inclusion of a lot of nodes data collectors.

IMUs are one of the main sensors involved in these techniques due to its ability to estimate some important parameters that can be used for indoor tracking. There are a huge variety of algorithms to combine measurements from different inertial sensor in order to obtain not only the position also the orientation. Furthermore, these sensors have been integrated in

PDR in order to detect activities and movement patterns. Actually, a combination of both, wireless signals and IMUs, are the state-of-the-art solutions for indoor positioning and tracking regarding the accuracy in the estimations.

In the other hand, image-based technologies are gaining an attractive interest due to the new techniques applied to image processing. In the recent years, Deep Learning techniques have changed the previous state of the art results in image processing techniques showing very accurate results in object detection and tracking. These solutions are the most common techniques in these moments when big computational resources are available for these tasks. The main interest is that the cost has been moved to the processing server and the quality of the camera is decreasing due to a lot of techniques can be applied to solve complex problems with no necessity of very high quality images.

Combining all these ideas, cameras and vision systems can collect information only in the visible range available but wireless signals and inertial sensors combined solutions can provide information out of the range of the cameras. Then, the combination of both systems can accurately track a person in big areas thanks to the high visible range of cameras and the availability of the wireless signals where cameras are not able to be useful.

Based on these premises Figure 4.1 shows the complete proposed architecture for accurate indoor tracking combining estimations from both, wireless/inertial sensors and cameras devices.

Figure 4.1: Proposed complete system architecture to fuse both wireless/inertial architecture and camera person tracking architecture.

One branch uses the wireless signals from several technologies and the inertial/magnetic information from a personal device allocate in the body. A Wireless Sensor Network (WSN) composed by several nodes is deployed along the monitored area and is able to collect the packets sent by the personal device. In addition, these packets contain the inertial/magnetic data from the sensors and each employed wireless technology in the nodes is able to collect the packets and extract the Received Signal Strength Indicator (RSSI) from the received packets. RSSI is the key parameter used in this work to estimate the position in the room. The WSN creates a new packet that is sent from each node to a main server that store all the information and contains all the algorithms to perform the processing.

The second branch takes as inputs the images from different types of cameras. This information is sent to the server that contains all the algorithms to get person trajectories from the images. The architecture is divided in three main parts, person detection, person tracking and person reidentification that will be described in detail in next chapters.

Finally, both branches are fused in a final module that takes the estimated trajectories from the sensor architecture and the image architecture, synchronize them and perform the fusion algorithm.

The devices employed in this work are presented in next paragraphs. It is important note that for the wireless/inertial system the main idea is combine technologies with low prize to put the complexity of the system in the server part. Due to the complexity of image-based algorithms using Deep Learning techniques the server is equipped with a powerful GPU. In this work Nvidia Titan RTX with 24GB of memory have been used for all the test related with deep learning algorithms. The idea behind the use of different camera types cameras is test the performance of different proposals in different image types that present concrete distortion patterns doing the detection part more complex.

Raspberry Pi v3 Model B devices were used as nodes, equipped with XBee S1 (protocol IEEE802.15.4, indoor range 30 meters, transmission power 0 dBm) Bluetooth USB Dongle PBU40 (protocol IEEE802.15.1 v4.0 BLE+EDR, indoor range around 10-20 meters, transmission power -3 dBm) and WiFi USB dongle TP-LINK TL-WN722N (protocol IEEE802.11 (b,g,n), indoor range around 40-80 meters, transmission power around 21 dBm), whereas the wearable was endowed with an Inertial Measurement Unit (IMU) Adafruit 9-DOF LSM9DS0 with accelerometer, gyroscope and magnetometer integrated, a XBee S1 transmitter, and the embedded WiFi and Bluetooth of Raspberry Pi Zero.

Figure 4.2: *Left:* Raspberry Pi 3 Model B. *Right:* Raspberry Pi Zero

Figure 4.3: Wireless devices and IMU+Magnetometer used in this work. From left to right: XBee S1, Bluetooth USB Dongle PBU40, WiFi USB dongle TP-LINK TL-WN722N, Adafruit 9-DOF LSM9DS0.

In the other hand, cameras system is compose by three cameras, one Normal Field of View (NFoV) camera with an aperture angle of 135° to cover extensive areas, a Fisheye camera with 180° field of view and a 360° camera.

Figure 4.4: Selected cameras in this work. *Left:* Vivotek FD8179-HV (NFoV camera). *Middle:* MOBOTIX c25 Hemispheric Indoor Camera (180° camera). *Right:* RICOH THETA V (360° camera).

Finally, the proposed scenario is presented in Figure 4.5 where 5 nodes for the wireless sensor network have been deployed and the 3 proposed types of cameras are allocated in different rooms. Nodes are presented colored circles and camera with colored squares. Yellow square correspond to the zenith camera, purple square is the 360° camera and brown square is the NFoV camera. The approximated widths and heights are also drawn in this figure.

Figure 4.5: Scenario where all the experiments were performed. Allocation of all the devices is drawn and approximated sizes of each room and corridors is presented.

Chapter 5

Proposed Wireless and Inertial IPS solutions

This chapter collect the main contributions to the research community using wireless signals and inertial sensors combining both in a multi-sensory system for accurate Indoor Positioning and Tracking. As introduction, the state-of-the-art systems are presented in two separate sections. Firstly, algorithms that use probabilistic and complex mathematical approaches are collected. Secondly, systems using Machine and Deep Learning techniques in some parts are drawn. The second part of this chapter is intended to present the main contributions of this in IPS presenting main problems to solve, proposed algorithms and results.

5.1 State of the art IPS solutions without learning algorithms using Wireless and Inertial sensors

5.1.1 Proximity Detection (Connectivity Based Positioning)

Proximity detection or connectivity based is one of the simplest positioning methods to implement. It provides symbolic relative location information. The position of a mobile client is determined by cell of origin (CoO) method with known position and limited range. When more than one beacon detects the mobile target, it simply forwards the position nearest where

the strongest signal is received. The accuracy of CoO relates to the density of beacon point deployment and signal range. This method is implemented with several wireless positioning technologies, in particular, the system running infrared radiation (IR), radio frequency identification (RFID), GSM (Cell-ID), Bluetooth, and custom radio devices [16, 17, 18].

5.1.2 Triangulation

Triangulation uses the geometric properties of triangles to determine the target location. It has two derivations: lateration and angulation. Techniques based on the measurement of the propagation-time system (e.g., TOA, RTOF, and TDOA) and RSS-based and received signal phase methods are called lateration techniques. The AoA estimation technique is also called an angulation technique. The main idea for this technique using this measurements is try to estimate the distances between the user device against several anchor nodes (Figure 5.1) [19, 20, 21].

Figure 5.1: Example of trilateration method where 3 nodes (Raspberry Pi as RPI) are used to estimate 3 distances (r_1 , r_2 , r_3) and the intersection between the circles using and optimization algorithm provide the final estimated position P.

5.1.3 Angle Based Method

The Angle of Arrival (AoA) technique determines the angle of arrival of the mobile signal coming from a known location at which it is received at multiple base stations. To estimate

position in a 2D dimension plane, AoA approach requires only two beacons. To improve accuracy, three beacons or more are used for location estimation (triangulation) [22, 23]. For finding direction, it requires highly directional antennas or antenna arrays. Geometric relationships can then be used to estimate the location of the intersection of two lines of bearing (LoBs) from the known reference points as shown in Figure 5.2.

Figure 5.2: Calculation of 2D position based on two antennas using AoA technique.

5.1.4 Time Based Methods

Lateralation/Trilateration/Multilateration terms refer to a position determined from distance measurements. Lateralation or trilateration determines the position of an object by measuring its distance from multiple reference points. Thus, it is also called range measurement technique. In trilateration, the "tri" says that at least three fixed points are necessary to determine a position. Techniques based on the measurement of the propagation-time system (for example, ToA, RTOF, and TDOA) and RSS-based and received signal phase methods are called lateralation techniques. The working method is the same as presented below in other techniques. Distances can be calculated based on system parameters, in this case the time of arrival. Using the combination of these distances a position can be obtained [24, 25, 26].

5.1.5 Signal Property Based Method

The majority of wireless localization systems compute the distance to the positioning device using either timing information or angle based. In both scenarios, they are influenced by the multipath effect. Due to this, the accuracy of estimated location can be decreased. The substitute method is to estimate the distance of unknown node to reference node from some sets of measuring units using the attenuation of emitted signal strength. This method can only be possible with radio signals. Mostly wireless localization systems positioning device using properties of the received signal, with received signal strength indicator (RSSI) being the most widely used signal-related feature. RSSI measurement estimations depend heavily on the environmental interference, and they are also nonlinear [27].

5.1.6 Dead Reckoning (DR)

Dead reckoning is the process of estimating known current position based on last determined position and incrementing that position based on known or estimated speeds over elapsed time (Figure 5.3). An inertial navigation system which provides very accurate directional information uses dead reckoning and is very widely applied [28, 29]. One of the disadvantages of dead reckoning is that the inaccuracy of the process is cumulative, so the deviation in the position fix grows with time. The reason is that new positions are calculated entirely from previous positions.

Figure 5.3: Dead reckoning example where step length is estimated along bearing angle to get the next position.

5.1.7 Map Matching (MM)

This method is based on the theory of pattern recognition which combines electronic map with locating information to obtain the real position of devices in a network. The use of maps is an efficient alternative to the installation of additional hardware. MM techniques include topological analyses, pattern recognition, or advanced techniques such as hierarchical fuzzy inference algorithms [30, 31].

5.1.8 Fingerprinting Based Indoor Localization

Most indoor localization approaches adopted fingerprint matching as the basic scheme of location determination. The main theme is to collect features of the scene (fingerprint) from the surrounding signatures at every location in the areas of interest and then build a fingerprint database. The location of an object is then determined by matching online measurement with the closed location against the database [32, 33, 34, 35].

This method does not require specialized hardware in either the mobile device or the receiving end nor is no time synchronization necessary between the stations. It may be implemented totally in software which can reduce complexity and cost significantly compared to angulation or purely time-based lateration systems.

The location fingerprinting also called a fingerprinting method consists of two phases. Phase 1 is the so-called calibration phase, offline phase, or training phase, and phase 2 is the localization phase or online phase (Figure 5.4).

In the first phase, a creation of radio maps for site survey where the positioning is supposed to work must be recorded. Basically, radio map is a database of spots at predefined points (coordinates) coupled with various radio signal characteristics, for example, RSSI, signal angles, or propagation time called signal fingerprints. Step by step, for every fingerprint, there must be a measurement that includes the information about all stations and their received signal strength (RSSI). When the localization system is operational, online phase, the mobile station measures signal properties at unknown spot. Then, the current measured signal strength values are compared for the best agreement with a database (radio map).

Figure 5.4: Fingerprinting procedure, phase 1 (fingerprinting generation and characterization) and phase 2 (new measurements matching)

The major drawback of the fingerprinting approach is the laborious and time-consuming calibration process. Furthermore, adding signal stations would challenge the ease of setup in fingerprinting. Beside this, the main challenge to the techniques based on location fingerprinting is sensitivity to environment changes such as object moving into the building (people, furniture, etc), diffraction, and reflection, which result in changes in signal propagation. To maintain the positioning accuracy, the calibration process should be periodically repeated to a recalculation of the predefined signal strength map. There exists some proposed method to solve this problem trying to automatically update the Fingerprinting database based on the environment conditions and the user location [36, 37].

Fingerprinting-based positioning algorithms using pattern recognition techniques are deterministic and probabilistic, K-nearest-neighbor (KNN), artificial neural networks, Bayesian inference, support vector machine (SVM), or their combinations [38].

5.1.9 IPS combining Positioning and Tracking

The inclusion of inertial sensors (accelerometers, gyroscopes and magnetometers) has increased the set of features that can be applied to prediction models, yielding to meaningful accuracy improvements for person movement monitoring. For the aforementioned techniques (RSSI and ToA), a Wireless Sensor Network (WSN) composed of a set of nodes deployed across the area under monitoring is utilized due to its low-consumption, self-organizing and easy deployment capabilities. Extensive research has been conducted in the literature to address the person tracking problem using RSSI [39, 40, 41, 42].

5.2 State of the art IPS solutions using learning algorithms and Wireless and Inertial sensors

5.2.1 Machine/Deep Learning using Wireless Signals

Using wireless signals the most common technique used in combination with Machine/Deep Learning algorithms is Fingerprinting. As previously was described Fingerprinting consist on the characterization of a big area collection several measurements of different concrete locations in a database. Using this database, a comparison is carry out in order to estimate the location closet to the device under localization.

Probabilistic methods exploit the RSSI variance for position estimation [43, 34, 35]. Recent approaches based on Machine/Deep Learning techniques are capable of employing the RSSI dataset to learn the optimal model that relates the measurements with the final position estimated. In [44, 45, 46], methods such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Neural Networks (NNs) are applied. Here, the time dependency of RSS values was not considered. In [47], KNN and a Kalman filter were used for localization performance using APs and other sensors. The accuracy was evaluated using mean square error. This work required extra hardware for data collection, and did not show system performances at different testing points. In [48], is proposed the use of a deep neural network (DNN) with linear discriminate analysis (LDA) to accurately localize by cleaning noise and utilizing the benefits of a large number of data in the dataset (Right part Figure 5.5).

In [49], a radial basis function network was used to estimate location, and a particle filter was used to track the user's path in an indoor environment. RFID and IR were used as the source of data from sensors, which creates a high cost as it requires additional hardware and requires offensive calibration processes. In [50], support vector regression (SVR) and an artificial neural network (ANN) were used for indoor localization. The data was generated by a propagation model in 2 GHz band. This work focused on locating APs in indoor environments, rather than localizing mobile users. In [51], authors used an ANN algorithm for indoor localization using ToA and AoA as data sources. However, this type of approach has operational complexity and is difficult to accurately localize because of lacks of LoS.

It is difficult to use traditional learning approaches for scalable localization in different complex and hierarchical environments because shallow learning has limitations of localization accuracy due to data size and the learning performance of the algorithms. It is highly affected by

different environmental factors, such as multipath fading and the attenuation of objects. Currently, DNN is applied to handle shallow learning problems by avoiding the needs of hand engineering for extracting feature representations, and easily learning high-level features from a large set of noise samples.

As discussed in [52], DNN shows immunity against signal fluctuations, noise effects, device dependency, and the elimination of time-consuming manual parameter tuning. According to [53], DNN helps to lower the workforce burden of localization. DNN can provide accurate Wi-Fi-based indoor localization due to the ability to learn signal fluctuations through time and environmental dynamicity because of its deeper functions that map the input to the output.

Figure 5.5: *Left:* Example of an Autoencoder network to learn to compress the information (RSSI from several nodes as input) and after use the encoded information to learn to predict the position. *Right:* Example of Deep Neural Network with an initial Encoder part to compress the information and a Classification/Regression part to predict location, building or floor depending on the available information.

In [54], a stacked denoising auto-encoder (SDA) (Left part Figure 5.5) was used to reduce the dimensions, and then the hidden Markov model (HMM) was applied to refine the localization. The root mean square (RMS) was compared with respect to different sample sizes, and shows that using SDA and HMM has better accuracy than using DNN and HMM alone. They show that it is possible to boost the localization time complexity using SDA and HMM. In [52], DNN was used for building and floor label positioning using RSS values. The proposed system was evaluated in terms of various epoch sizes. In [53], a DNN system was used for building and floor classification on the publicly-available UJIIndoorLoc dataset. This work used a stacked auto-encoder for feature reduction. Researchers reached the conclusion that the deep learning paradigm can provide a global location recognition solution from WiFi data

at a significantly reduced effort in hierarchical problem types. Nevertheless, the work did not consider localization accuracy evaluation at each testing point.

In [46], channel state information (CSI) was used for indoor localization. Researchers compared the accuracy of the proposed system performance in laboratories and living rooms and found that the system performs differently, even in environments which are nearly the same. This work has no way to determine the performance of the system in different scenarios and testing points. In [55], a feedforward neural network (FFNN) classifier was used for multi-building and multi-floor indoor localization. The experimental scenario was performed in a building, while their system was evaluated through a simulation in a multi-building setting. However, in real world situations, it is very difficult to evaluate a system through datasets collected from different environments in different approaches. In [56], building-level classification was done by selecting the strongest RSS values from the measured fingerprints. Here, it is difficult for the system to discriminate buildings whenever there is similar signal strength.

In [57], a deep-learning-based indoor fingerprinting system using channel state information (CSI) was used for indoor localization. In [58], MLP was used for indoor localization using the mitigation of the ground reflection effect for position estimation. The localization has been done through AoA data sources that have LoS problems. It describes the relationship between the RSSI signals sent by the APs and the distance between such an AP and a set of mobile device users. In [59], authors used an artificial synaptic network, a MLP network, and a generalized radial basis function network through ToA measurements to localize sensors rather than localizing WiFi users. In [60], MLP and KNN were compared using data collected from a $17 \times 10 m^2$ area of the environment. The system performance was evaluated as a range-free approach in each mobile node by comparing the system performance at each RSS value obtained.

Convolutional neural networks were applied to extract the features of spatio metrics in accordance with convolutional layers [61, 62, 63, 64]. Although the spatio metrics may be analyzed by neural networks and convolutional neural networks, these methods cannot provide the solutions of temporal data analyses. Therefore, this study applies recurrent neural networks to analyze the temporal data for improving the accuracies of estimation locations.

5.2.2 Fingerprinting recovering and update

Although the method of building a fingerprint database for localization is very efficient, the construction of fingerprint database usually requires significant manpower and time to collect data. Many researchers have put forward solutions to this problem.

It is demonstrated that the method of Fingerprinting is very efficient for indoor positioning and tracking but normally exists a huge manpower effort to create and maintain the complete database. Some researchers have addressed this problem trying to automatize the procedure generating automatically new measurements to complete or modify the collected database. In [65], authors propose a novel approach based on compressive sensing for recovering absent fingerprints. Experimental results showed that with only half of the fingerprints, this approach can recover all the fingerprint information with reduced error. [66] they propose a Gradient Fingerprinting which leverages a more stable RSSI gradient building a gradient-based fingerprint map (Gmap) by comparing absolute RSSI values at nearby positions.

Authors in [67] present a fingerprint-based localization system that achieves extremely low overhead in fingerprint map construction and maintenance. They treat a scan from any reference locations as an input to adjust a large portion of the fingerprint map. In [68], a continuous update of the fingerprinting database is done by taking current accurate location of the person and estimate the new RSSI values combining the semivariogram information and a technique named Kriging. ACMI is presented in [69] where using the fingerprint database as a knowledge base the algorithm refines a positioning result via path matching, during its runtime. Work presented in [70] draw a novel a method for detecting WiFi APs to form WiFi fingerprints from the signals collected by ZigBee interfaces.

Authors in [71] propose a system that automatically generate deterministic and probabilistic accurate radio maps for WLANs localization systems. The work collected in [72] a hybrid generative/discriminative semi-supervised learning algorithm is proposed that utilizes a large number of unlabeled samples to supplement a small number of labeled samples. In [73] a novel method to adapt the temporal radio maps for indoor location estimation by offsetting the variational environmental factors using data mining techniques and reference points is presented. Authors in [74] exploit the spatial correlation structure of the fingerprints and use the framework of Matrix Completion to build complete training maps from a small number of random sample fingerprints.

The authors of [75] proposed the Enriched Training Database. It is a web-service that enables

the management and storage of training fingerprints, and it also has an additional enriching functionality. The user can automatically generate virtual fingerprints based on propagation modeling in the virtual training points by using the enriching functionality. They also proposed a new method of training fingerprinting locations that eliminates the burden of manually defining training points and covers areas that do not have enough density to train fingerprints. In [76], the authors proposed a novel method to construct a fingerprint database with full fingerprints by using the radio propagation model. For limitations of the indoor dynamic measurement, the authors in [77] proposed a GPR-based radio map construction. It uses realistic and virtual indoor dynamic measurement data.

Figure 5.6: AF-DCGAN architecture [5]. It is composed by a Generative Adversarial Network which is able to generate Channel State Indicator (CSI) maps in indoor environments.

To expand the fingerprint database with reduced human effort, authors in [5] present a method to increase the number of training data at each point based on Generative Adversarial Network (GAN). First, they transform CSI data collected at each reference point into amplitude feature maps. Then, through resolution transformation, each amplitude feature map is transformed into an image of the same pixel to construct the initial fingerprint database. Since the initial database is constructed by mapping the amplitude feature maps of the reference points, it contains the location information of all reference points. Amplitude Feature Deep

Convolutional Generative Adversarial Network (AF-DCGAN) model is proposed as well as the corresponding training algorithm to generate images similar to the original amplitude feature maps. The generated amplitude feature maps are merged into the initial fingerprint database to obtain an expanded one.

5.2.3 Machine/Deep Learning in INS and Dead Reckoning

In one hand, people will encounter different situations while walking, and the types of activities will also change, accordingly the model parameters will also not be the same. The fixed model parameters are usually only useful for a single state of motion and will no longer be suitable for the complex and mixed athletic situations. Adjusting the model parameters adaptively according to the type of motion is a method used by some researchers to solve the motion trajectory estimation under these situations.

In the other hand, there are some restrictions on the current PDR systems in many application scenarios. For example, devices are restricted to being on the front side of the body stably without changing poses [78, 79], otherwise, the pedestrian's position cannot be obtained, which greatly increases the limitation of localization and cannot satisfy the portability requirement of pedestrians.

The most common pedestrian movements are walking and running. Based on this, some of the literature considers other types of motion, such as upstairs, downstairs, side stepping, standing, etc. Considering some more detailed cases it can be included walking and running forward, backward and sideways, respectively. In [80] it is established a database of 14 different daily activities, which, besides walking and running, also includes jumping, falling, etc. Running was divided into the activities of running, sprinting, jogging, while crouching and climbing were also considered in the literature [81]. Due to the differences in environment, region, country and so on, the types of human activities vary widely, and it is impossible to cover and classify all types of human movements. However, by classifying some common types of activities, more prior knowledge was obtained for PDR systems, thus creating a potential for higher trajectory estimation accuracy and solving some of the problems of the device location.

Before the classification is performed, the features of the input data need to be extracted first. The original output data of the inertial measurement units generally include triaxial acceleration, triaxial angular velocity, and triaxial magnetic field strength. A single frame of

output data can only represent one feature dimension. In order to obtain more data features, it is necessary to analyze other characteristics of the output data. For example, as is described in [82], commonly used statistical features including mean, standard deviation, variance, mean absolute deviation, root mean square (RMS), interquartile range, correlation coefficient, etc. On the other hand, authors in [83] divided the data features into time domain features and frequency domain features. The time domain features include the mean, maximum, minimum, variance, and energy of acceleration and angular velocity; the frequency domain feature is the Fourier transformation of statistical features in the time domain.

After the feature is extracted, the next step is to use the features to train a specific machine learning model for motion classification. There are many methods for pattern classifications. The main classification methods used are Probabilistic neural network classifier [82], Support vector machine classifier [84, 83], Multi-layer perceptron [83], Decision trees [85, 86], Hidden Markov model [87], Boosting algorithms [88] and recurrent neural networks [89].

Authors in citePei2012UsingLB presented an indoor navigation solution by combining physical behavior recognition with wireless positioning. They compared 27 features extracted from the smartphone sensors and the least squares support vector machine (LS-SVM) classification algorithm is used to detect eight behavior patterns that commonly occur in indoor navigation. In [90] it is defined six kinds of common movements in indoor navigation and utilized an artificial neural network (ANN) for the classifier.

To relieve the burden of designing and selecting features, authors of [91] first applied stacked denoising autoencoders for locomotion activity recognition. The proposed method was independent of the expert knowledge and greatly reduced the work of manual feature design, by the selection and combination of various sensors, the results shown that the classification performance by multiple sensors can be better than that by only accelerometers.

Accelerometers, gyroscopes and magnetometers are used to obtain the yaw angle of the device, and the yaw angle is regarded as the heading of the pedestrian. In [92] authors proposed an indoor location method, which enhanced the heading estimation of PDR with the received signal strength indicator (RSSI) from the Wi-Fi access points (APs). When a long straight trajectory has been detected by PDR, the headings of PDR can be corrected by the headings derived from the RSSI data. In [93] it is developed a PCA-based method to infer the orientation of mobile device carried in a pocket from the acceleration signal.

Of course, not all PDR algorithms need to classify the motions, because adding the motion classification step will undoubtedly increase the complexity of the algorithms. Differences in

the lectotype, number, and location of sensors may cause the trained models not to perform well. However, without considering other aids, in order to achieve the trajectory estimation under complex environments and mixed motions, it can indeed increase the accuracy of the estimations by adding motion classification algorithms.

In the case where the precision of the inertial sensors cannot be significantly improved, the more prior knowledge about the movements there is, the better the accuracies of the algorithms will be. Without any prior identifications and classifications for the motions, there is currently not such a robust algorithm that uses only micro inertial sensors and does not rely on other aids to accurately estimate the trajectories for any motions. Therefore, it is a good choice to accurately classify and recognize the motions in real time to achieve more accurate trajectory estimations under complex motion conditions.

Figure 5.7: Deepsense architecture [6]. It is composed by a combination of Convolutional Neural Networks and Recurrent Neural Network to work with data in both temporal and spatial domain.

Finally, DeepSense [6] is the first architecture that possesses the capability for both modeling temporal relationships and fusing multimodal sensor inputs using CNNs and RNNs. It also contains specifically designed structures to exploit local interactions in sensor inputs.

5.3 Proposed sensors system architectures

During this work Fingerprinting technique was adopted as the based method for positioning estimation. Why Fingerprinting?. This technique allows characterize the environment recording previously measurements in a predefined time in different locations. This amount of data is suitable for the application of probabilistic methods or learning algorithms. Three main architectures have been proposed for this purpose and are described in the following sections [94, 68].

5.3.1 Fingerprinting definition

Complex indoor environments have been considered, where harsh signal propagation conditions present a challenge for appropriate characterization. For this aiming, it is assumed an indoor scenario with a set $\mathbf{v} = \{v_1, v_2, \dots, v_N\}$ of Access Points (AP) located at Cartesian coordinates $\mathbf{x}_j = (x_j, y_j) \in \mathbb{R}^2$ creating a Wireless Sensor Network (WSN). Furthermore, let l be the number of sensors that each node is equipped with. In this work, sensors from three wireless technologies are considered: IEEE802.11.g/WI-FI [95], IEEE802.15.1/Bluetooth v4.0 [96], IEEE802.15.4/xbee and IEEE802.15.4/CC2420 [97]. These sensors will obtain the Received Signal Strength Indicator (RSSI), denoted by $\mathbf{z}$, from the monitored target. Consequently, it is denoted by $\mathbf{z}^l(v_i(\mathbf{x}_j))$ the RSSI measurements from the sensor l gathered by node v_i at location $\mathbf{x}_j$.

- $\mathbf{z}$ is assumed to be an independent and identically distributed (i.i.d.) Random Variable.
- Every node v_i is able to gather a set of RSSI measurements ($\mathbf{z}$) from the aforementioned technologies l . Notice that it is possible to obtain $\mathbf{z}^l \rightarrow \emptyset$
- T is defined as the measurement period. All data obtained in this time window will be assumed to be synchronized.
- Notation $\mathbf{z}^l(v_i(\mathbf{x}_j))$ describes the measures sensed by node v_i at a location $\mathbf{x}_j$ from fingerprint stage.

Figure 5.8: *Left:* This figure represents the real monitored scenario and the cells distribution for fingerprinting. Five nodes equipped with the selected wireless technologies are deployed across the area (*circles*). Doors are drawn in red and furniture in gray respectively. *Right:* the reference coordinates system for inertial and magnetic sensors as well as the allocation of devices employed are illustrated.

The orientation of the wearable device attached to the body is presented in the right side of Figure 5.8. The device is allocated on the person hip and axes (X,Y,Z) in the figure match the IMU axes.

Large variance of measurements gathered is one of the most critical problems associated to RSSI estimation, especially for indoor environments, even in case of static objects [98]. An example of this variance is shown in Figure 5.9 where a large amount of RSSI measurements are taken from a static object equipped with several sensors. Fingerprinting technique allows to obtain a proper scenario calibration by splitting it into a set of cells $\mathbf{c} = \{c_1, c_2, \dots, c_\ell\}$ with geometrical center $\mathbf{x}_j = (x_j, y_j) \in \mathbb{R}^2$ respectively. The larger the number of cells ℓ is, the higher the spatial resolution is. Measures are taken from each node v_i in the center of every cell $\mathbf{x}_j$ for a predefined time interval τ .

Figure 5.9: Example of RSSI variance boxplots in the collected Fingerprinting database is presented for each node in different cells selected for a particular wireless technology (XBee). Variance in measurements is large, yielding to significant fluctuations in the signal distribution over a particular location (i.e. a cell in the monitored room).

As a result of this procedure, a distribution of the RSSI is captured in every cell k for every technology l and every node v_i . For this purpose, it is defined $\mathbf{F}$ matrix that contains the fingerprinting distributions. Therefore, $\mathbf{F}^l(v_i(\mathbf{x}_j))$ represents the set of fingerprint measurements gathered for cell $\mathbf{x}_j \in \mathbb{R}^2 \wedge c_j \in \mathbf{c}$ by sensor technology l in node $v_i \in \mathbf{v}$. Moreover, $\mathbf{F}$ matrix is the input of the TPE process described in next subsection.

A typical indoor scenario is characterized in Figure 5.8 (scenario used in the experiments performed in this section) where the width and depth are denoted by W, D respectively. Additionally, a set of cells $\mathbf{c} = \{\mathbf{p}_l | l = 1, \dots, L\}$ with geometrical center at $\mathbf{p}_l = (x_l, y_l) \in \mathbb{R}^2$ is defined, where c_i is a particular cell. The distance from a cell to another is marked by $d(c_i, c_j)$ using euclidean distance as metric.

It is assumed that a/several individuals is/are moving in the scenario wearing mobile devices endowed with at least on of the aforementioned technologies (smartphone, wearable, etc).

5.3.2 Fingerprint calculation

Aforementioned factors affect the quality of RSSI measurements yielding to large variance in the power level detected by every node, which is critical when the detection is performed in an indoor scenario as the one shown in Figure 5.8. Fingerprinting is employed to characterize the monitored scenario by splitting it into a set of cells and obtaining RSSI measurements at every cell for a predefined time. However, some important aspects such as the number of cells, the point of interest for cell measurements and the coverage area are generally neglected.

5.3.2.1 RSSI Variance

Initially, it can be argued that the larger the number of cells is, the higher characterization resolution achieved is. However, if the number of cells is large, the statistics could be similar between each other, increasing the estimation errors for Fingerprinting cells with identical behaviors or very close among them. Consequently, it can be also stated that the larger the number of fingerprinting cells is, the higher probability of error in estimation is.

Therefore, an analysis is required in order to select the cell size to prevent the overlap between RSSI measurements collected in the nearest cells. In practice, RSSI values overlap for different distances in multiple cells due to the propagation conditions. It can be inferred that the more distances with the same RSSI reading overlapping, the higher error probability in position estimation.

An example of this statement is depicted in Figure 5.9 where a set of four cells and all the RSSI measurements collected by all nodes are presented. A single node has a coverage range constrained for every particular technology, and any of these covers the whole area. As a consequence, unexpected effects such as a non-uniform data distribution from sensors around this position or even lack of coverage makes difficult to fit a proper propagation pattern for each cell. As an example, for Cell 8, it is shown that measurements collected from Node 2 and 3 (the closest nodes in our deployment) are larger than measurements collected from Node 1 and 5, whereas no measurements from Node 4 (the furthest node with more walls in between).

5.3.2.2 Wireless Technology Signal Attenuation

Wireless technologies used in fingerprinting have different signal distributions across the distance. There exist several approaches for indoor propagation modeling where the signal attenuation is relevant to define the cell size. As an example, cases where signal attenuation in a cell is very low (i.e. almost flat) have a similar distribution giving to unreliable estimations for this area. Based on this statement, the nodes must be allocated accordingly to the area under monitoring avoiding to have these nearly from each other. This effect is mitigated in the presence of multiple wireless technologies as its particular propagation patterns allow to distinguish the locations by fusing the data from each one.

5.3.2.3 Sampling Time per Cell and Synchronization

The data collection period for every cell is relevant due to the need of gathering enough RSSI measurements to obtain a representative sample. The number of packets forwarded depends on the processing capabilities of transmitting devices. In order to synchronize the measurements collected, it is assumed that the transmitter device is able to embed the timestamp to the measures collected. Fixed length slicing windows of $40ms$ were created to collect measurements that are assumed to be synchronized. In the tests performed in this work, the sampling frequency for the considered technologies is up to 25 samples per second and the sampling time per cell was adjusted to 5 minutes.

5.3.2.4 Object or Person Tracking

The absorption of people must be considered for human tracking applications. Several works have addressed this problem trying to model the human absorption [99]. In this work, empirical tests were performed to determine that the average attenuation in detection due to human absorption. In this work, measurements along several locations and different orientations with the device attached to the body and the device fixed without human interactions have been collected. In addition, some measurements with 3-6 persons moving along the monitored area were considered to emulate interference. The mean of the information gathered was used to establish an attenuation equal to 4 dBm.

5.3.2.5 Orientation

The antenna orientation effect has also implications on the statistical RSSI positioning algorithms. In general, such assumption remains only in the symmetric region of the antenna. However, when communications are presented in a non-symmetric region, which is generally the 3D case, there is a considerable variation of RSSI values. Moreover, the radiation pattern is not perfectly symmetric in all directions, having an impact on the signal propagation. An experiment rotating a 802.15.4 wireless device in the same plane, as explained in [100], has confirmed that the orientation of the device is a significant attenuation factor in the RSSI value. In Figure 5.10, it can be observed an example of the measurements collected in a cell rotating over all the angles and representing the RSSI measurements for a single node in four directions. There exists a large variability in the measurements when the device is pointing to the node (-30 dBm) or to the opposite side (-40 dBm).

Figure 5.10: An example of different orientations in the same cell is presented. The orientations have been concentrated in four angle ranges. The large variability in the measurements depends heavily on the orientation of the device in the cell.

5.3.3 Probabilistic sensor fusion architecture

In this section a first approach for positioning and tracking using Fingerprinting technique in combination with probabilistic methods is described. The complete architecture consists on two main parts, a target position estimation part that extract the estimated position based on Kernel Density Estimation (KDE). The second part take the advantage of Kalman Filter to perform the tracking along several wireless technologies. Two proposal of sensor fusion to combine the estimation for each technology are also drawn in this subsection.

5.3.3.1 Target Position Estimation (TPE)

There exist several works in the literature for position estimation based on Fingerprint statistics [101, 40, 70, 102]. Most of these works apply deterministic techniques to characterize the RSSI distribution for each cell. In [101], the K-Nearest Neighbors (KNN) technique is employed to triangulate the target position adopting Euclidean distance as metric. In [40], a comparison of various distance metrics is carried out in an indoor building giving as result that Mahalanobis distance outperforms Manhattan and Euclidean-Distance results. Furthermore, in [70] a weighted K-Nearest Neighbors approach is presented. Moreover, in [102] it is proposed a complete overview of several deterministic methods that include all the aforementioned metrics. However, taking into consideration a large set of measures collected for every cell in Fingerprint stage, a deterministic approach can neglect the wide distribution of measurements per cell, therefore a probabilistic approach can be considered for the estimation. In particular, Bayes theorem-based methods can be applied for position estimation as follows:

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z})} \quad (5.1)$$

where $p(\mathbf{x}|\mathbf{z})$ is the conditional probability and $p(\mathbf{x}, \mathbf{z})$ is the joint probability.

Kernel Density Estimation (KDE) is a non-parametric technique employed to obtain the Probability Density Function (PDF from now on) of a random variable with independent and identically distributed samples [103, 104]. KDE estimates the likelihood that Fingerprinting distributions match the measures in the region of a Kernel function. Therefore, the distance

is fitted with Kernel functions, denoted by $K(\cdot)$, that must fulfill the following conditions:

$$K(-\mathbf{x}) = K(\mathbf{x}) \quad (5.2)$$

$$\int K(\mathbf{x}) d\mathbf{x} = 1 \quad (5.3)$$

equation (5.2) states that Kernel functions must be non-negative. Moreover, equation (5.3) implies that Kernel functions, as well as the PDF must be unbiased (therefore be integrable to one). If both conditions are satisfied, the PDF can be calculated in its general form by the following function:

$$p(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{\mathbb{V}} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right) \quad (5.4)$$

where n is the set of samples. Furthermore, $\mathbb{V}$ is a normalization parameter of the Kernel volume and h_n is a smoothing parameter also known as kernel bandwidth. The goal pursued is to estimate a non-parametric function that fits the RSSI values distribution better than known (parametric) distributions. Specifically, equation (5.4) can be applied to our fingerprinting by employing histograms to smooth the measurements distribution. Therefore, the maximum likelihood function can be expressed as:

$$p(\mathbf{z}(v_i)|c_j) = \frac{1}{nh} \sum_{k=1}^n K\left(\frac{\mathbf{z}(v_i) - \mathbf{z}_k(v_i(\mathbf{x}_j))}{h_n}\right) \quad (5.5)$$

where n is the number of measurements gathered in every fingerprinting cell $j \in \mathbf{c}$ by the node $v_i \in \mathbf{v}$, $\mathbf{z}(v_i)$ is the set of RSSI measures gathered by node v_i in a window time T . In addition, $\mathbf{z}(v_i(\mathbf{x}_j))$ is the set of RSSI values gathered by node v_i in fingerprint cell located at x_j . Due to the nature of the data modelled, a non-parametric method is chosen to calculate the joint density probability function. In particular, a modified version of KDE known as Nadaraya-Watson Kernel regression is used [105]. This method is applied because of its appropriateness when there is no prior knowledge of the relationship between the variables under study since these estimators are only based on either smoothing or regression functions.

There are other non-parametric options that could have been employed such as the Priestley-Chao and Gasser-Müller smoothers or the K-th Nearest-Neighbor (K-NN) weights [106]. However, Priestly-Chao and Gasser-Müller strict boundary bias problems make Nadaraya-Watson method the most suitable estimator for RSSI-based applications. In fact, it has been demonstrated that Nadaraya-Watson variance is up to a 50% lesser than the other methods [106]. However, K-NN smoothing can attain similar results to the ones obtained by using kernel estimation when an appropriate bandwidth h parameter is adjusted.

According to this method, the joint probability function can be estimated as:

$$p(\mathbf{x}, \mathbf{z}(v_i)) = \frac{1}{\ell} \sum_{k=1}^{\ell} \frac{1}{(h_{\mathbf{x}})^2 (h_n)^N} K\left(\frac{d(\mathbf{x}_k, \mathbf{x})}{h_{\mathbf{x}}}\right) K\left(\frac{\mathbf{z}(v_i) - \mathbb{E}[\mathbf{z}(v_i(\mathbf{x}_k))]}{h_n}\right)$$

where $d(\mathbf{x}_k, \mathbf{x})$ represents the distance from every fingerprinting cell center to the estimated point and $h_{\mathbf{x}}, h_n$ are the bandwidth parameters respectively.

The most common Kernel functions used to fit the PDFs in equation (5.6) are the Gaussian and Exponential:

$$K(\mathbf{x}) = (2\pi)^{-\frac{N}{2}} |\mathbf{C}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{x}' \mathbf{C}^{-1} \mathbf{x}\right) \quad (5.6)$$

$$K(\mathbf{x}) = \frac{1}{2} \exp(-\|\mathbf{x}\|) \quad (5.7)$$

The former is the Gaussian Kernel (5.6), where N is the number of nodes considered for estimation, whereas $\mathbf{C}$ represents the covariance matrix which is a diagonal matrix with the covariances of the N node measurements. The equation (5.7) is the Exponential Kernel where $\|\mathbf{x}\|$ represents the euclidean distance.

Applying the Minimum Mean Square Error (MMSE) criteria over the conditional density probability:

$$\hat{\mathbf{x}}_{MMSE} = E(\mathbf{x} | \mathbf{z}(v_i(\mathbf{x}))) = \int_C \mathbf{x} p(\mathbf{x} | \mathbf{z}(v_i(\mathbf{x}))) dx \quad (5.8)$$

it is obtained that:

$$\hat{\mathbf{x}} = \sum_{k=1}^{\ell} w_k \mathbf{x}_k \quad (5.9)$$

where $\mathbf{x}_k \in \mathbb{R}^2 \wedge k \in \mathbf{c}$. The target position $\hat{\mathbf{x}}$ is calculated as the weighted sum of the estimations for every cell using the Kernel function. The weights w are determined by:

$$w_k = \frac{K\left(\frac{\mathbf{z}(v_i) - \mathbb{E}[\mathbf{z}_k(v_i(\mathbf{x}_k))]}{h_n}\right)}{\sum_{j=1}^{\ell} K\left(\frac{\mathbf{z}(v_i) - \mathbb{E}[\mathbf{z}_j(v_i(\mathbf{x}_j))]}{h_n}\right)} \quad (5.10)$$

The numerator contains the current Kernel for cell $k \in \mathbf{c}$ and the denominator collects the sum of all Kernel values for every cell. The Nadaraya-Watson Kernel regression is the MMSE estimator of $\mathbf{x}$.

Using this algorithm, we get the estimates of the position starting from the measures of RSSI while the wireless devices are moving in the room. The problem inherent in these estimates is that presenting a high noise due to the effects of the movement and by the person who moves around the room. These estimates do not allow to obtain the final path followed by the devices properly. By this, is makes necessary perform a filtered that allow estimate the appropriate path that reducing the noise presented in the estimates.

The input of to Equation 5.10 will be the RSSi mean at each cell and the deviation (depending on the Kernel used) and the current collected RSSI at timestamp t . Finally, the TPE output is the estimated value $\hat{\mathbf{x}}$ which is assumed to be static in T . Moreover, in next subsection, the tracking algorithm to obtain the complete route is drawn.

5.3.3.2 Target Tracking Process (TTP)

Tracking stage is intended to minimize the effect of noise in $\hat{\mathbf{x}}$ by filtering outliers giving as a result the peaks-free object route. Several methods have been proposed for this purpose,

however in this work the well-known Kalman Filter (KF) [107, 108] is employed. This filter provides a recursive solution based on Least Squares Method.

KF assumes the system to be described by a linear stochastic model, where the error associated to the system as well as the additional information incorporated are presented as normal distribution variables with zero mean and variance σ^2 .

A complete illustration of Kalman Filter is drawn in figure (5.11). This filter performs the Best Linear Unbiased Estimator (B.L.U.E) of a system state using information from the previous estimations. KF consists of two stages: (a) time update (left) and (b) measurement update (right) are related. The former is in charge of projecting future estimators of the current state and error covariance. The latter will predict new state estimations.

Figure 5.11: Kalman Filter Algorithm steps for prediction and estimation.

One of the most interesting KF features is the ability to predict the state of a system, even if the nature of the system modelled is unknown. The *Time Update (prediction)* estimates the

state $\mathbf{x}$ according to the following equation:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}_k u_k + \eta_k \quad (5.11)$$

where $\eta \sim \mathcal{N}(0, \sigma)$ is the process noise, $\mathbf{A}$ matrix relates the k state with the $k+1$ time state. Furthermore, $\mathbf{B}$ is the control input associated to u , which is the system input. Furthermore, $\mathbf{P}$ collects the deviation in the estimates that generally is initialized as a diagonal matrix and $\mathbf{Q}$ is the covariance matrix.

Moreover, *Measurement update (prediction)* stage in figure 5.11 is described as follows:

(1): The matrix $\mathbf{G}$, called Kalman Gain or blend factor, adjusts the trade-off between the estimation and the error measurement.

(2): Equation to calculate the posteriori state $\mathbf{x}_{k+1}$ as a linear combination of the a priori $\mathbf{x}_k$ estimator and the weighted difference between current observations $\mathbf{z}_k$ and a measurement prediction $\mathbf{H}$.

(3): Update the error covariance based on the Kalman Gain. Moreover, the observation $\mathbf{z} \in \mathbb{R}^2$ contains the spatial coordinates of the position :

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + v_k \quad (5.12)$$

Finally, matrix $\mathbf{H}$ relates the state with the measure $\mathbf{z}_k$ where $\mathbf{z}_k$ is the TPE output and $v \sim \mathcal{N}(0, \sigma)$ the measurement noise respectively.

In this work, a Modified version of the Kalman Filter (MKF) [109] is applied. The reason to use MKF is that the model considered in this system is linear. Consequently, a linear version of KF taking into consideration velocity and acceleration parameters represents a good trade-off between complexity (i.e Extended Kalman Filter) and performance (velocity and acceleration improve the estimation results). In addition, Extended Kalman Filter does not provide the optimal solution in case of wrong initial state inputs, which is highly probable in the scenarios considered in this work [110].

Therefore, MKF algorithm considers the speed axis v_x and v_y variables for the state $\mathbf{x}$ in addition to the position vector $\hat{\mathbf{x}} \rightarrow (\hat{x}, \hat{y})$ of the tracked target. These variables represent

an advantage in terms of estimation accuracy since speed endows the system with a more realistic movement modelling. Therefore, the system can be expressed as follows:

$$\mathbf{x} = [x, y, v_x, v_y]' \quad (5.13)$$

$\mathbf{x}$ is the initial state composed by axis positions and speed. Consequently, matrix $\mathbf{A}$ will be defined as:

$$\begin{cases} x_{k+1} = x_k + v_x dt \\ x_{k+1} = y_k + v_y dt \\ v_{x_{k+1}} = v_{x_k} \\ v_{y_{k+1}} = v_{y_k} \end{cases} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.14)$$

Additionally, $\mathbf{H}$ remains as the general Kalman formulation whereas $\mathbf{B}$ and u_k will be modified as follows:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2}dt^2 \\ \frac{1}{2}dt^2 \\ dt \\ dt \end{bmatrix} \sigma_a^2 \quad (5.15)$$

Main modifications presenting the movement information and are included in different matrices of the process, where $\sigma_a^2 = u_k$ is the acceleration parameter. Moreover, $\mathbf{R}$ matrix contains the error variance of TPE estimation:

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \quad (5.16)$$

where σ_x^2, σ_y^2 represent the error variance estimation in x and y respectively. Finally, covariance matrix $\mathbf{Q}$ represents the uncertainty in the estimation process incorporating the entire set of

parameters and variance σ_q^2 :

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4}dt^4 & 0 & \frac{1}{2}dt^3 & 0 \\ 0 & \frac{1}{4}dt^4 & 0 & \frac{1}{2}dt^3 \\ \frac{1}{2}dt^3 & 0 & dt^2 & 0 \\ 0 & \frac{1}{2}dt^3 & 0 & dt^2 \end{bmatrix} \sigma_q^2 \quad (5.17)$$

The value of σ_p in this matrix represents the uncertainty of the process. Other values are dependent on measuring range and represent the states of update with information from the acceleration in the x and y position values with their respective speeds in each axis. Finally, these values will decrease as the filter performs estimates converging to a final value of estimated error.

5.3.3.3 Multi-Sensor Processing Architectures

Two main multi-sensor schemes are considered in this work: (1) Track-to-track fusion and (2) Kalman Sensor Group Architecture. Both architecture patterns are shown in Figure 5.12 and Figure 5.13 respectively. In the former, processing stages (TPE and TTP) are performed separately for each sensor to achieve high-level inferences (routes) that are subsequently fused. Nonetheless, in the latter scheme the sensor measurements are combined in the TTP stage. Finally, in order to update automatically the fingerprinting database in time a final stage perform a final processing that will be presented in the final subsection of this section.

- **Track-To-Track Fusion Architecture TTTF**

This architecture pattern is based on fusing the routes obtained by each sensor l separately. The algorithm proposed applies the Maximum Likelihood Estimator (MLE) to the probability density function of the available routes [111]. For this purpose, $\mathfrak{R}$ is defined to be the set of routes. Additionally, it is assumed that every route is composed by ϖ number of steps $\mathfrak{R}^l = \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_\varpi\}$. Therefore, the probability density function can be described as follows:

$$p(\mathfrak{R}^l) = \prod_{k=1}^{\varpi} \prod_{n \in l_k} \frac{1}{\sqrt{(2\pi)^d |\mathbf{P}_n|}} e^{-\frac{1}{2}(\hat{\mathbf{x}}_n - \bar{\mathbf{x}}_k)' \mathbf{P}_n^{-1} (\hat{\mathbf{x}}_n - \bar{\mathbf{x}}_k)} \quad (5.18)$$

Figure 5.12: *Track-to-track Fusion Architecture:* Separate processing stages for each sensor to achieve high-level inferences that are subsequently fused.

where $\mathbf{P}$ and $\mathbf{x}$ are the outputs from *TTP* described in section 5.3.3.2. Note that $\hat{\mathbf{x}}$ describes the position estimated from previous stages, whereas $\bar{\mathbf{x}}$ denotes the final output of the Fusion stage. From equation (5.18), it has been demonstrated [111] that MLE can be obtained by minimizing the result of $\mathbf{x}$ and setting the gradient equal to 0. Therefore, the resulting log-likelihood function is:

$$\Lambda(\varpi, \{\mathcal{R}^l\}, \{\mathbf{x}_k\}_{k=1}^{\kappa}) = \frac{1}{2} \sum_{k=1}^{\varpi} \sum_{n \in l_k} \left(\ln((2\pi)^{\mathfrak{d}} |\mathbf{P}_n|) + (\hat{\mathbf{x}}_n - \bar{\mathbf{x}}_k)' \mathbf{P}_n^{-1} (\hat{\mathbf{x}}_n - \bar{\mathbf{x}}_k) \right) \quad (5.19)$$

where $\mathfrak{d}$ is the dimension. Moreover, the state estimation for every location can be calculated

as follows:

$$\bar{\mathbf{x}}_k = \left(\sum_{n \in l_k} \mathbf{P}_n^{-1} \right)^{-1} \sum_{n \in l_k} \mathbf{P}_n^{-1} \hat{\mathbf{x}}_n \quad (5.20)$$

Note that, both the estimation $\bar{\mathbf{x}}_k$ and $\mathbf{E}(\bar{\mathbf{x}}_k) = \bar{\mathbf{x}}_k$ are unbiased. Finally, the covariance of the estimate, denoted as Σ , is given by:

$$\Sigma_{\varpi} = \left(\sum_{n \in l_k} \mathbf{P}_n^{-1} \right)^{-1} \quad (5.21)$$

It can be shown that this result is the Kalman Filter update equation applied to each local route in each partition. Further details can be found in [111].

- **Kalman Sensor Group Fusion Architecture KSGF**

The second architecture presented is the Kalman Sensor Group Fusion (KSGF) which exploits the Kalman filter properties for multi-sensor data fusion [112]. KSGF method directly incorporates the position estimations of each sensor in a single Kalman Filter. However, the system must be properly conditioned to obtain an accurate route estimation.

General KF system equations were described in section 5.3.3.2. However, some modifications must be introduced in the filter to support multi-sensor functionalities. In concrete, update matrix $\mathbf{H}$, described in equation (5.15), must be adapted to incorporate the sensor inputs:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (5.22)$$

Figure 5.13: *Kalman Sensor Group Fusion Architecture:* direct sensor data combination.

Furthermore, $\mathbf{R}$ matrix must include sensor error covariance measurements. These values are appended to the main diagonal:

$$\mathbf{R} = \begin{bmatrix} \sigma_{x_1}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{y_1}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{x_2}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{y_2}^2 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{x_l}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{y_l}^2 \end{bmatrix} \quad (5.23)$$

Finally, the observations vector $\mathbf{z}$ is now extended to collect the x and y coordinates estimated

from each sensor:

$$\mathbf{z} = [\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2, \dots, \hat{x}_l, \hat{y}_l]'$$
 (5.24)

Taking into consideration the aforementioned modifications, the KF can be performed as described in *TTP*. As a result, the fused route will be provided by the KSGF. Both architecture patterns can be employed for multi-sensor data fusion. The computational complexity of the architectures proposed has been addressed [113]. In this work, both patterns are compared in terms of execution time. KSGF computing time grows exponentially as the number of sensors increases. In addition, TTTF scheme outperforms KSGF in terms of estimation accuracy. Therefore, in this paper TTTF architecture is adopted for estimation process.

5.3.3.4 Alpha-Beta Filter for noise reduction

In order to minimize the noise and outliers' effect that can arise as result of multi-sensor stage, an additional filtering stage is applied. There exist several proposals for routes smoothing, however in this paper we use the well-known alpha-beta filter [114]. The reasons to employ the alpha-beta filter are the lower complexity, the lesser computational cost and that system model details are not required. Furthermore, taking into consideration that the effect of outliers' is much lesser in the presence of low noisy estimations, it has been shown that the alpha-beta filter estimations performance is similar to Kalman Filters [115].

This filter is a particular case of the KF described in previous sections and consequently it assumes that a system can be adequately approximated by a two states model [114, 116].

Similarly to equation (5.13), initial states of spatial coordinates and axis speeds are considered. x, y, v_x, v_y . Therefore, the position prediction can be obtained as follows:

$$\hat{x}_{k+1} = \hat{x}_k + \hat{v}_{x_k} t ; \hat{y}_{k+1} = \hat{y}_k + \hat{v}_{y_k} t$$
 (5.25)

Additionally, error measurements can be calculated from predicted states by the subtraction of the fusion algorithm $\hat{x}_m, \hat{y}_m$ from the predicted position:

$$e_x = \hat{x}_m - \hat{x}_{k+1} ; e_y = \hat{y}_m - \hat{y}_{k+1}$$
 (5.26)

The final position and velocity for next algorithm iterations are obtained as follows:

$$\hat{x}_k = \hat{x}_{k+1} + \alpha e_x ; \hat{y}_k = \hat{y}_{k+1} + \alpha e_y \quad (5.27)$$

$$\hat{v}_{x_k} = \hat{v}_{x_k} + \frac{\beta}{\Delta t} e_x ; \hat{v}_{y_k} = \hat{v}_{y_k} + \frac{\beta}{\Delta t} e_y \quad (5.28)$$

where α and β are the filter control parameters. In order to guarantee filter stability and convergence, these values must fulfill the following conditions:

$$0 < \alpha < 1; 0 < \beta \leq 2; 0 < 4 - 2\alpha - \beta \quad (5.29)$$

Additionally, there is a direct relation between alpha-beta control parameters:

$$\beta = \frac{\alpha^2}{2 - \alpha} \quad (5.30)$$

5.3.3.5 Experimental results

In this section, the experiments performed to assess this proposed architecture and the results obtained in comparison with existing methods are detailed. Firstly, the TTP performance is assessed by comparing this architecture to other methods that employ fingerprint technique. The cell splitting is drawn in Figure 5.14. Four nodes at the top of the corners in the room was used. The experiment is run for a total of 1000 different routes. Additionally, every route $\mathfrak{R}^l = \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_{\varpi=90}\}$ comprises 90 steps. Measurements were carried out with 4 wireless technologies, Bluetooth, Wifi, XBee and CC2420 (similar to XBee) reading simultaneously in a period $\tau = 5 \text{ minutes}$ per cell.

Figure 5.14: Cell splitting for experiments using the proposed probabilistic architecture.

The average error for every method is detailed in table 5.1. System results are significantly better than deterministic methods. In addition, the results with the proposed system are better than Gaussian Kernel methods except for WiFi sensors. This is probably due to two aspects: (a) distances considered are not long enough to appreciate WiFi signal attenuation and (b) the histogram distributions are similar to Gaussian functions.

Table 5.1: Average error comparison of MUFAF with several methods expressed in centimeters.

Positioning Method	XBee	CC2420	WiFi	Bluetooth
Euclidean Distance [40]	110.71	150.367	122.72	150.57
Mahalanobis Distance [40]	108.33	140.05	112.12	142.58
K-NN [70]	105.87	139.58	157.42	161.29
Gaussian Kernel [101]	67.27	61.92	63.30	170.34
Exponential Kernel (This Work)	63.05	59.49	79.73	137.15

Furthermore, results obtained for the aforementioned sensor technologies are shown in Figure 5.15. In this figure it can be observed that Bluetooth technology errors in estimation are larger than the other technologies. The reasons for this result can be the low power transmission of Bluetooth devices and interference's. Conversely, 802.15.4 based technologies perform with a mean error under $65cm$.

Figure 5.15: Average error comparison of route estimation for every technology (no fusion applied) using Fingerprinting technique, Kernel Density Estimation and Kalman Filter.

Moreover, the architecture patterns described in section 5.3.3.3 were also analyzed. The experiment consists of a set of 300 random routes with 90 measurements per route. Comparison results of both architectures are drawn in Figure 5.16. It can be appreciated that TTTF outperforms KSGF in terms of error. TTTF involves to run TTP separately to fuse routes obtained for each sensor showing better tracking results than performing KSGF. As a consequence, TTTF allows optimization of KF parameters in every specific sensor. However, the average error for each architecture pattern is shown in table 5.2, where the difference is 16.16 cm which demonstrates that TTTF outperform KSGF.

Figure 5.16: Average error using both fusion architectures in path estimation for an experiment composed by 300 routes with 90 steps per route.

Table 5.2: Comparison of architecture error for 300 routes

	Track-to-track	Sensor Group
Average error cm	49.68	65.84

A particular example of routes and MUFAF performance is shown in figure 5.17. Green dotted line describes the real route whereas the red line describes the path estimated for both: TTTF (left side) and KSGF (right side).

Figure 5.17: (*left*) Route estimation by using Track-To-Track Fusion Architecture and (*right*) route estimation by using Kalman Sensor Group Fusion Architecture.

In the view of results obtained, it can be concluded that the Track-to-Track Fusion strategy outperforms Kalman Sensor Group Fusion. Finally, in general terms, technologies employed in this work can be fused to improve the accuracy of estimations.

5.3.4 ANN sensor fusion architecture

The second proposal for accurate indoor tracking takes the advantage of learning techniques such as Machine and Deep Learning. In this case, a feature vector composed by important parameters that have effect in tracking have been estimated using an Inertial Measurement Unit composed by Accelerometer, Gyroscope and Magnetometer. The combination of sensors and RSSI from different wireless technologies is the input to an artificial neural network that will learn to estimate the position. After the estimation, a filtering based on the movement parameters is applied and finally a particle filter is applied over the estimations to get the final trajectory.

5.3.4.1 SWiBluX System Architecture

Fingerprinting is a method for positioning in particular times, rather than a continuous tracking technique over time. Both, positioning and tracking methods are presented in the architecture drawn in Figure 5.18. SwiBlux system architecture is composed by several stages: First, a synchronization process over the measurements is needed to create the feature vector. Second, a Deep Learning approach is used to perform the fingerprinting estimation. Third, a Gaussian Filter is proposed to reduce the outliers impact on the estimations. A weighted combination of the estimations after the filter provides an estimated position which is inserted into a Particle Filter to track the person based on a realistic movement model. The use of additional sensors and the proper fusion of diverse sources improves the detection accuracy. The velocity estimation algorithm provides information about how fast a person is moving in the monitored area and the Madgwick [117] filter is used for orientation estimation to improve the final results in some stages of the algorithm.

5.3.4.2 Fingerprinting Position Estimation

In this work, Machine Learning techniques have been compared with Deep Learning approaches, where the output of this procedure is a probability for all the cells (at center) position based on the feature vector inserted to the algorithm.

$$[RSSI_{1,\lambda}, RSSI_{2,\lambda}, \dots, RSSI_{I,\lambda}] \quad (5.31)$$

Data in this format can be analyzed employing different methods such as Distance-based Metrics (i.e. Euclidean, Mahalanobis, Spearman), K-Nearest Neighbors including its versions (W-KNN) or more sophisticated algorithms such as Kernel Density Estimators. Nonetheless, these methods require some statistical information including both the first and second moments of the distribution as well as the percentiles of RSSI signals in each cell.

Machine/Deep Learning methods can exploit the entire dataset to learn more complex patterns using all the available information. The first step is to gather the RSSI measurements for each cell by every node and create a custom feature vector as shown in the Example 5.31. The resulting dataset containing these vectors for each cell is used to train the mentioned algorithms.

In Artificial Neural Networks (ANN) training stage, neurons are taught to identify specific incoming signals and whether activate or not for a particular pattern/output. In related works the information of the fingerprint in combination with the multiple modalities (from inertial sensors) were employed to train an ANN to learn the most likely cell locations of a target and generalize the probabilities of associated propagation effects. Once the ANN has been trained on samples of the known database, it is able to predict by detecting similar patterns for future input data even when these have not been seen previously. In this work, as input of ANN, the following feature vector (FV in Equation 5.32) to tackle the Fingerprinting problems in an indoor scenario is presented:

$$FV = [RSSI_{1,X}, RSSI_{2,X}, \dots, RSSI_{I,X}, RSSI_{1,B}, \\ RSSI_{2,B}, \dots, RSSI_{I,B}, RSSI_{1,W}, RSSI_{2,W}, \dots, \\ RSSI_{I,W}, Yaw] \quad (5.32)$$

where $\{X, B, W\}$ indicates $X = XBee$, $B = Bluetooth$ and $W = WiFi$, I represents the number of nodes deployed and Yaw is the Heading angle estimated by the Madgwick Filter.

Figure 5.18: SwiBlux system architecture is composed by several stages: *First*, a synchronization process and a Deep Learning approach is used to perform the fingerprinting estimation (green boxes). *Second*, a Gaussian Filter is proposed to reduce outliers in the estimations (orange boxes). *Third*, the step, velocity estimation and absolute orientation (blue boxes). *Fourth*, Particle Filter tracks the person based on a realistic movement model giving the final position (yellow boxes).

This vector includes all available wireless technologies which implies a synchronization stage that allows to employ data from each node deployed for each wireless signal. For instance, in a certain time slot, an XBee packet could have not being received due to the attenuation, whereas the rest of technologies collected properly the RSSI value. Moreover, in spite the Bluetooth signal is weak and the coverage range is not large, the estimation is possible since the rest of technologies have a larger coverage and might be available in cases where Bluetooth is not.

Furthermore, the Yaw/Heading angle introduced in the vector provides relevant information for the orientated RSSI in the cell as shown in Figure 5.10. The orientation affects the measurements gathered, and these changes are registered during the Fingerprinting creation by a person wearing the device (to take in account body absorption impact) rotating continuously in the cell reference. The corresponding time to collect data on each cell is fixed to 5 minutes. In such period, the continuous rotation during the RSSI gathering allows to cope with multiple possible events to statistically characterize the RSSI cell distribution and the propagation behavior in each direction.

Figure 5.19: Deep Neural Network composed of: an input layer with size the selected feature vector, two hidden layers and the last layer with size equal to the number of cells with a final Softmax activation.

The selected feature vector is used to construct the complete Fingerprinting dataset for ANN training and testing. An ANN with two hidden layers is selected as network structure with a final *Softmax* activation (Figure 5.19). The activation function selected in the hidden layers is ReLu. By using two hidden layers, it is possible to represent an arbitrary decision boundary with rational activation function that can approximate any smooth mapping to any accuracy.

Additional layers can learn more complex representations. Nonetheless, there is a trade-off between architecture layers (Deepness) and the computational complexity of the proposed system. In this work, significant improvements were not attained by increasing the number of layer (> 2) whereas the training process was longer and the delay in estimation was perceived.

Figure 5.20 presents the SWiBluX complete training architecture. Synchronization between the wireless sensors employed is needed and the estimations of Yaw value from inertial and magnetic sensors during database collection step. The proposed scheme can be observed as a classification problem faced with an ANN where RSSI and inertial measures are inserted as input and the cell estimated is provided as output (classes).

Several approaches can be used to select the number of parameters to construct the network [118]. The method to choose both the number of hidden layers and neurons is the *Try and Error Method* that is characterized by repeated, varied attempts which are continued until success or until the agent stops trying. A Forward propagation approach was selected to tune the network parameters as it works better in models with medium/low size of input data as the one presented in this work.

Figure 5.20: General Scheme of the Training process. *From top to bottom:* Fingerprint creation from the N Network nodes for each particular technology in addition to the orientation values (Yaw) is depicted. The length of the feature vector inserted into the Neural Network is $(I(nodes) \times \lambda(technologies)) + 1(Yaw)$.

Moreover, weights initialization has played a crucial role in this work due to the fact that *zeros initialization* is heavily penalized during the training process by not learning at all during epochs with more complex optimization algorithms or learning quickly with simpler optimization algorithms. Therefore, *He initialization* [119] was used in this work.

Since the input data is not very sparse, some methods such as *SGD* (Stochastic Gradient Descent), *NAG* (Nesterov Accelerated Gradient) and *Momentum* are preferred. In the complicated task of selecting the optimal learning rate, *SDG* provided the best results on the dataset created in this work and it was chosen consequently.

The learning rate was fixed to a small value of 10^{-3} to find a local or global minima during the optimization by avoiding large jumps over the loss function surface. Further, no overfitting was observed in the experiments, therefore, no regularization method has been included during training.

For the training process, the data was split into two subsets, the training and test sets. The ANN model can learn more efficiently with low-range numbers as input. The proposed feature vector was normalized into zero-to-one values due to the fact that all these parameters are always between a range of -92 and -25 for RSSI values and yaw/heading value between $0 - 360$ degrees respectively.

Once the feature is passed, the back-propagation optimizes the weights to minimize the *Categorical Cross Entropy Loss Function* using the selected optimization algorithm.

5.3.4.3 Gaussian Outliers Filtering and Weighted Position Estimation

Machine/Deep Learning based positioning algorithms provide each cell probabilities in the Fingerprinting distribution. In a real scenario, a person moving across the monitored area can stay at different locations besides the cell center. The last layer in the Neural Network usually performs the *Softmax* activation function to obtain a vector with the cell probabilities for a particular input feature vector. However, in case of wrong estimations, a stochastic error is introduced in the next step of the algorithm using only the center of the cell as estimation.

Figure 5.21: Example of the Gaussian Outliers Filtering. The bidimensional Gaussian is centered at the previous estimated position and the deviation is the velocity calculated at the current stage. The cell estimated by the Neural Network, in this case, is wrong so the filter applies a low probability to it. In this manner, the larger probability is assigned to the nearest cells (the ones that the filter covers).

Layer previous to the *Softmax* activation contains the probabilities for each cell of being selected for position estimation. The proposed system removes the *Softmax* activation layer (at testing stage) and applies a Probability Density Estimation (PDE) filter to separate the unlikely cells (i.e. it is not possible that a person jumps several cells from a previous instant). Additionally, this filter smooths the probabilities of the most likely cells to reach a fine tuning based on the last estimation.

The two-dimensional Gaussian Function (Equation 5.33) is applied over the outputs of the last layer in the ANN. The key concept is to multiply the outputs of this layer with the ones generated by the two dimensional Gaussian Function. General statistical tools, the *mean* and *deviation* on both spatial axis (x, y) are required to generate this function. The mean is obtained using the last position estimated (previous stage) by the Tracking Filter. The deviation is not a constant value and depends on the person movement. Therefore, an adaptive value is chosen to guarantee that the Gaussian Filter width fits with the person

movement at each moment.

$$f_X(x_f, y_f) = \frac{1}{(2\pi)^{1/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu) \right) \quad (5.33)$$

Person velocity is the key parameter used as the Gaussian Filtering deviation due to its ability to adapt to person changes. To cope with multiple cases, the Gaussian Filter can be wide-spread allowing medium-far cells to be considered into the position estimation when the velocity is high or conversely, to employ narrow filters for slow velocity. This procedure changes the probabilities and provides a normalized vector $\mathbf{p}(\mathbf{Gauss})$. Afterwards, a weighted combination of the cells based on these probabilities yield the final estimated location entering to the Tracking Filter as shown in Figure 5.21.

$$x_g = \sum_{l=1}^L \frac{p(Gauss)_i}{\sum p(Gauss)_i} x_l \quad y_g = \sum_{l=1}^L \frac{p(Gauss)_i}{\sum p(Gauss)_i} y_l \quad (5.34)$$

Equation 5.34 x_g, y_g shows the estimated position before the tracking filter and $p(Gauss)_i$ represents the probability of a cell (with center at location (x_l, y_l)) that can be candidate to the weighted combination.

5.3.4.4 Step detector and velocity estimation

In this work, the accelerometer measurements are used to extract information about the walking/standing state to estimate the velocity based on the signal analysis and patterns. The inertial sensors provide information about the acceleration in the three coordinates axes. An example of measurements gathered during a normal walking behavior are presented in left part of Figure 5.22. Root Mean Square (RMS) estimation from the three accelerometer axes for the signal analysis was used for processing the raw acceleration data and it is calculated as follows 5.35:

$$RMS_{acc} = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (5.35)$$

Figure 5.22: *Left:* Three axis accelerometer measurements. Regarding the proposed referenced coordinate system it can be shown that the Y axis measurements are around $1g$ representing the gravity force ($9.8m/s^2$) in this axis. *Center:* Example of peaks/valleys detection. Red dots represent valleys and green dots represent peaks. It can be observed that the amplitude needs to be large between peaks/valleys to detect them correctly. *Right:* Estimated velocity after Monte Carlo Simulation to get the parameters for the movement model and filtering the output.

To detect steps, maximum (maxima) and minimum (minima) acceleration peaks can be determined by using top and bottom thresholds respectively. However, this method does not work properly due to the large variation of both maxima and minima amplitudes during the walking phases. This issue can cause either lack of peaks or multiple peaks in a single step period. Conversely, the maxima peaks have less amplitude variations than the minimum. To reduce or even eliminate completely this undesirable effect, a peaks/valleys detector with an adaptive threshold is employed. Such algorithm is known as Adaptive Step Jerk Threshold [120] and it is capable to modify the threshold to filter peaks or valleys with no sufficient amplitude or consecutive false peaks/valleys. Using this information, person step length can be determined.

There are empirical equations to obtain the velocity based on these parameters [121, 122, 123]. However, most of these approaches do not fit properly the velocity model analyzed in this paper. The performance comparison allowed to establish that the best results are obtained using [121]. Equation 5.36 describes the velocity model based on the output of the peaks/valleys algorithm:

$$L_{step} = K_{conv} \sqrt[4]{A_{max} - A_{min}} \quad (5.36)$$

where A_{max} is a peak, A_{min} represents the consecutive valleys and K_{conv} is a constant for unit conversion. It is observed that the equation does not adjust this model perfectly due to the fourth square root and the unknown value for the conversion parameter, although this model presents a similar behavior. A Monte Carlo simulation was performed over the root value and the conversion parameter based on different walking measurements of diverse individuals minimizing the error between the real and the estimated velocity.

The last step is to filter the final estimated velocity values. An example of peaks/valley detection along with the estimated velocity by the algorithm over these values are presented in the center and right part of Figure 5.22.

5.3.4.5 Absolute orientation estimation

Common approaches in orientation measurement utilize low cost sensors that lack of accuracy. Nonetheless, the information of accelerometers, gyroscopes and magnetometers can be fused to attain reliable orientation estimations. Three often used choices include the complementary filter [107], the Kalman Filter [108] and the Mahony/Madgwick Filter [109, 111]. In SWiBluX, the Madgwick Filter was chosen due to its good performance when applied over non high precision sensors, low computational complexity and real-time processing.

There exist calibration algorithms to reduce the errors produced by sensors high sensitivity [112]. In this work, the ellipsoid calibration algorithm [113] was selected because of its good trade-off between precision and data required for calibration. This algorithm collects measurements from all the possible sensor positions in the space to correct the bias as well as concentrate the corrected values in a normalized sphere of radius equal to one (Figure 5.23).

Once the calibration procedure has finished, the absolute orientation can be estimated. This filter uses a quaternion representation, allowing accelerometer and magnetometer data to be used in an analytically derived and optimized gradient-descent algorithm to compute the direction of the gyroscope measurement error as a quaternion derivative.

$$Yaw = \arctan2 \left(\frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2} \right) \quad (5.37)$$

$$Pitch = -\arcsin \left(2(q_1q_3 - q_0q_2) \right) \quad (5.38)$$

$$Roll = \arctan2 \left(\frac{2(q_0q_1 + q_2q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2} \right) \quad (5.39)$$

The filter output is the estimated quaternion (four dimensional vector) based on the actual sensors measurements. The quaternion vector is employed to calculate the Yaw/Heading, Pitch and Roll as shown in Equations 5.37 , 5.38 and 5.39 respectively.

Figure 5.23: Magnetometer calibration results. *Left:* uncalibrated normalized data points, *right:* calibrated normalized data points after ellipsoid calibration.

5.3.4.6 Filtering Tracking with Particle Filter

The tracking algorithm is applied over the estimated position from the previous stage to filter and reduce estimation noise as well as to adjust the positions to an human movement model. It is applied to reduce the fast non-natural movement errors and to fit the model to a like-human movement as shown in Equation 5.40:

$$\begin{aligned} x_{pf} &= x_p + \hat{v} \cos(\theta)dt \\ y_{pf} &= y_p + \hat{v} \sin(\theta)dt \end{aligned} \quad (5.40)$$

where x_p, y_p are the coordinates of the previous positions estimated, $\hat{v}$ is the estimated velocity and θ is the yaw/heading angle. The employed tracking algorithm is the well-known Particle Filter [124] which is a common choice for applications such as robotics by estimating position, velocity and heading angle in a similar manner to SWiBluX.

The idea behind Particle Filters is that any Probability Density Function (PDF from now on) can be represented as a set of samples (particles). Each particle has a set of values for the state variables representing arbitrary distributions, making it appropriate for non Gaussian, multi-modal PDFs. Therefore, it is feasible to find an approximate representation of a complex model (any arbitrary PDF) rather than an exact representation of a simplified model (Gaussian).

To start the algorithm, an initial belief state, denoted as $p(x_0)$, which is a rough estimation of the PDF is required. For localization problems where no previous knowledge of the distribution is available, it is possible to scatter particles over the whole map. In this work, the particles are scattered around the first estimated position in the first stage with a Gaussian distribution centered at that position with deviation equal to twice the estimated velocity. This is done to cover a wide area of the map in case that the previous estimation might not be reliable. For each time step, the algorithm loop contains three main phases: *prediction*, *update*, and *resampling*.

In the *prediction* step, each particle is displaced according to the motion model. The resulting position from the motion model, will be somewhere in that cloud of particles. The resulting distribution of particles approximates the prior distribution.

During the *update* step, weights are assigned to the sensor measurements according to the probability of observing the sensor measurements from that particle's state to each particle. The weights are normalized and its sum is equal to 1 (Equations 5.41 and 5.42) to guarantee unbiased decisions.

$$\mathbf{L}_{xy} = \frac{1}{\sqrt{2\pi}\sigma} \exp \left(\frac{-\sqrt{(\mathbf{x}_p - x_g)^2 + (\mathbf{y}_p - y_g)^2}}{2\sigma^2} \right) \quad (5.41)$$

$$\mathbf{w} = \frac{\mathbf{w} \cdot \mathbf{L}_{xy}}{\sum_{i=1}^M w^{(i)} \cdot l_{xy}^{(i)}} \quad (5.42)$$

where L_{xy} represents the particle importance with respect to the previous estimation and M is the number of particles.

In the *resampling* step, a new set of particles is chosen maintaining the ones with the largest weights. Therefore, unlikely particles (small weights) at the fringe are not chosen whereas the most likely particles (near the center of the cloud) are replicated. As a result, the high-probability region has a high density and properly represents $p(x)$ (the posterior distribution). There are several methods for weights resampling [125], but the Systematic Resample [126] is selected as it faces with the event of particles wandering away from the point and its weight tends (or is equal) to zero. Consequently, a small number of particles implies a reduced number of particles contributing to the approximation of the distribution. Moreover, in order to determine whether resampling is needed or not, the coefficient of variation statistic [127] can be calculated using the following equation 5.43:

$$CV = \frac{1}{M} \sum_{i=1}^M (Mw^{(i)} - 1)^2 \quad (5.43)$$

The effective sample size can be calculated as shown in Equation 5.44. This parameter describes how many particles have an appreciable weight. Therefore, to check if resampling is necessary, the effective sample size can be tested against the number of particles as follows:

$$ESS = \frac{M}{1 + CV} \quad \text{Resample if } ESS < \tau \cdot M \quad (5.44)$$

where τ is a threshold between 0 and 1. In this work, this value is selected as 0.5 which means that not resampling is requested in case the half part of the particles exceeds the ESS value.

Finally, the new estimated position is obtained based on the calculated weights and the particles after the prediction step as Equation 5.45 indicates:

$$x_f = \sum_{i=1}^M \mathbf{w} \cdot p_x \quad y_f = \sum_{i=1}^M \mathbf{w} \cdot p_y \quad (5.45)$$

where (p_x, p_y) are the x and y position of each particle respectively and M is the number of particles.

5.3.4.7 Results on Position Estimation and Tracking

This section shows results on location estimation with the most common methods in Fingerprinting positioning, comparing with the results achievable by different algorithms of both Machine Learning (ML) and Deep Learning (DL) techniques. Furthermore, results on the use of the Gaussian Outliers Filtering and the Weighted Combination of the outputs of this filter after ML /DL methods demonstrates an accuracy improvement in the position estimation. The final location of the person wearing the device is provided by using Particle Filter reducing the final estimated error.

The configuration of the scenario is shown in Figure 5.8 where cells and device allocation are presented. Five nodes $\mathbf{v} = v_1, \dots, v_5$ are deployed and each node is equipped with three wireless sensors technologies $\Lambda = 3$: IEEE802.15.1 (Bluetooth), IEEE802.11.g (WiFi) and IEEE802.15.4 receiver (XBee).

The monitored area has been split into 70 cells of different size varying between 150×150 cm to 240×240 cm. The system is able to collect up to 25 RSSI measurements per second per technology. The collected dataset contains RSSI measurements from each employed wireless technology and Yaw values during 5 minutes in each cell. To consider scenario changes, the measurements have been collected in five different rounds of 1 minute per cell. After the synchronization stage, the database contains $7.5k$ feature vectors per cell.

To calculate the accuracy, it is necessary to know where the person is located during a complete route in the monitored area (a.k.a ground-truth). Several routes were collected to get the error in the estimations, comparing the estimated position at each algorithm step and getting the error from the estimated position over the real position using the Euclidean distance metric.

Figure 5.24 shows the accuracy obtained using several methods for fingerprinting estimation taken different approaches for the feature vector in two cases. Accuracy is interpreted as the mean euclidean distance between the estimated position and ground-truth. In the first case, the fingerprinting stage is applied whereas the second case contains the accuracy after applying the proposed Gaussian Outliers Filtering in combination with the Weighted Combination of the probabilities for each cell. These feature vectors contain all the possible combinations of the three technologies employed.

Figure 5.24: Estimation accuracy in position estimation using several methods with different feature vectors in two main cases: The first case (*left*) corresponds to perform the normal Fingerprinting estimation based on the selected method, the second (*right*) shows the results after applying the GOF and the weighted combination proposed in this paper. For horizontal axis X , B and W denote XBee, Bluetooth and WiFi technologies respectively.

Based on these two figures:

- It can be observed that all methods reach a larger accuracy in the estimations after the application of the GOF followed by the weighted cell combination based on the resulted probabilities.
- Different feature vectors present diverse results. As an example, Bluetooth technology alone has the largest error in all methods as well as it is clear that the combination of technologies outperforms single technology-based estimation. WiFi technology provides a more robust RSSI (with less variance). Thus, all methods work well using this wireless technology in the feature vector.
- The use of the proposed feature vector outperforms the rest of combinations due to the introduction of the Yaw parameter which provides information about the orientation, improving the pattern learning process by Machine/Deep Learning algorithms.

- Deterministic and Probabilistic methods have the worst results compared to learning algorithms. The method including the feature vector proposed in this work cannot be applied to distance methods such as Euclidean, Mahalanobis and W-KNN as enough relevant information cannot be obtained from the Yaw to be applied.
- The Deep Neural Network achieves the largest accuracy results in both schemas. Deep Learning algorithms can learn more complex and non-linear patterns that are limited in the Machine Learning methods.

The velocity model selects the parameters that reach the best fitting to the real measurements with the proposed system. A Monte Carlo simulation over a range of possible values was used to search for the best parameters in several velocity measurements. Finally, the mean of the parameters obtained at each evaluation over real measurements and the final equation are (Figure 5.25, Equation 5.46):

$$K_{conv} = 0.81 \quad r_{exp} \approx \frac{1}{3} \quad L_{step} = 0.81(A_{max} - A_{min})^{\frac{1}{3}} \quad (5.46)$$

Figure 5.25: Monte Carlo simulation over a range of possible values for adjust the the step length model to the detected peaks and valleys by the adaptive step jerk algorithm.

These parameters adjust adequately to the proposed method for velocity estimation in different cases when a person is walking slowly, rapidly or even when a person is stopped.

Table 5.3: Average error in position estimation (centimeters) for several methods using fingerprint at different stages of the proposed system architecture with the proposed feature vector

Positioning Algorithm \ Proposed System Stage	Fingerprinting	Fingerprinting After Gaussian Outliers Filtering	Fingerprinting + GOF + Weighted Combination	Particle Filter
Deterministic and Probabilistic Common Methods				
Euclidean Distance	259.56	NA	NA	156.34
Mahalanobis Distance	230.56	NA	NA	145.21
Weighted-KNN	210.56	NA	NA	135.67
Gaussian Kernel	184.56	147.78	120.54	102.53
Exponential Kernel	191.32	153.62	132.02	111.36
Machine Learning Methods				
KNN Classifier	135.99	111.21	101.77	87.08
Logistic Regression Classifier	133.76	111.32	99.89	82.32
Support Vector Machine	104.54	90.78	85.67	78.65
Decision Tree Classifier	97.87	90.62	84.98	75.76
Gaussian Naive Bayes Classifier	89.43	82.22	78.45	69.91
Random Forest	88.59	77.79	74.76	62.48
Gradient Boosting	85.32	74.65	71.78	60.56
Deep Learning Methods				
Deep Neural Network	80.67	68.72	61.88	54.41

Analyzing the results presented in Table 5.3, it can be observed a significant improvement after each stage in the system architecture:

- GOF improves the final accuracy by filtering critical outliers that can degrade the performance of the final estimations.
- The Weighted Combination based on the resulted probabilities enhances the estimated position selection, not always placed at the center of the cell, thus reducing the final error.
- Particle Filter tracking algorithm in combination with the estimated parameters (Yaw, velocity) achieves a satisfactory performance with better estimated positions and reducing the average error up to 15 cm.
- Figure 5.26 represents an example of a person tracked and the particles used by the Particle Filter at three different moments. The number of particles is decreasing over the time according to the selected threshold by the *ESS* parameter. It can be shown that the inclusion of both Velocity and Yaw provide meaningful information to the filter achieving high accuracy in the estimations.

Figure 5.26: Particles (blue dots) managed by the Particle Filter at different moments during the tracking. The red dots represent the estimated position at each moment based on the particles whereas blue line depicts the final path estimated using the proposed architecture.

Table 5.4: Different Neural Networks configuration presented with several assessment parameters

Eval. Parameters	Precision	Recall	F1 score	Time (ms)
One Hidden Layer 256 neurons				
Neural Network 1	78%	80%	79%	0.896
Two Hidden Layers 256, 64 neurons				
Neural Network 2	89%	88%	87%	1.235
Two Hidden Layers 512, 256 neurons				
Neural Network 3	90%	81%	85%	2.146
Three Hidden Layers 512, 256, 64 neurons				
Neural Network 4	93%	72%	83%	3.689

The ANN has been tested with different configurations looking for high precision while avoiding overfitting. Furthermore, the selection was subject to estimate quickly the position in order to maintain estimations in (near) real time. Four experiments were performed and are presented in Table 5.4. The number of epochs in all configurations is 100. The batch size

has been reduced to 64 to learn quickly reducing the overfitting at early stages of the algorithm. Early stopping is used to finish the training if the precision tends to decrease due to overfitting.

To ensure a correct dataset splitting in training and validation sets, at learning phase each network was trained ten times with random splitting at each training and the final precision/recall/F1 values are the mean of these ten experiments. The training data size after each splitting is the 85% of the total dataset collected and the test/validation data has the 15% of the total measurements.

Figure 5.27: Training Loss and Accuracy for Training and Validation sets in two hidden layers (256, 64 neurons respectively) neural network configuration.

- Configuration four (in table 5.4) reaches the best precision but the recall is not large. This is a symptom of the starting of overfitting since the number of neurons in the network layers is large regarding the low size of the input feature vector. It means that the ANN learns quickly the patterns and after some epochs, it *overfits* the final estimations reducing the performance.
- The first configuration presented a short processing time and an acceptable precision, although it is lower than precision obtained by configurations two and three.

- Configurations two and three attain the best precision and recall results. However, configuration two has been selected for the experiments due to its reduced the computation time.

Configuration two has been selected for the experiments regarding the precision of the estimations and the computation time (a simple schema of this network is presented in Figure 5.19). Both precision and loss during training are depicted in Figure 5.27. The loss value decreases after each iteration in the training set and the loss in the validation set follows the same decreasing pattern. The precision grows up in training as depicted in the figure but early stopping ended the training process at epoch 49 as the next iteration precision and loss do not decrease.

Final important comments about the results can be that the Fingerprint dataset was constructed taken into account the Fingerprint parameters for diverse situations, orientation and human presence were considered. The proposed feature vector with the inclusion of the Yaw/Heading information enhances ML/DL training accuracy up to 15%. Final results in Fingerprinting estimated position using learning algorithms outperform the common deterministic and probabilistic methods up to 45 – 50%. The proposed Gaussian Outliers Filter increase the positioning accuracy around 10 – 15%. Particle Filter improve final tracking estimation around a 25% in contrast with only positioning.

5.3.5 Recurrent Neural Networks for continuous indoor tracking

The last system architecture proposal is presented below. It consists on a Recurrent Neural Network which takes as input a new proposed feature vector similar to the presented in the previous architecture with the ANN. The network will learn to track the position working with sequential data (tracking data composed by RSSI and movement parameters). All the steps involved to train the RNN will be described in next sections.

5.3.5.1 RNN system architecture

Recurrent models are a powerful solution to work with data in time and to perform time series analysis. An extensive research exists to apply these networks to different types of problems. One extensive topic is text analysis and prediction. Another topic is focus on sounds and

music analysis and generation. Finally, RNNs are commonly used to predict new values knowing previous information. In this work RNNs are used for this last application. Taking as input previous position, measured RSSI values and movement information (velocity and orientation) the task is learn a model that can create from this input data a realistic human movement model based on wireless signals to predict next person position.

The proposed input data or feature vector (FV) is compose by a total of eight parameters (Equation 5.47):

$$FV1 = [x_{t-1}, y_{t-1}, RSSI_1, RSSI_2, \dots, RSSI_N, \theta, \nu] \quad (5.47)$$

Where x_{t-1} and y_{t-1} are the previous estimated position to the actual timestamp t . $RSSI_n$ is the collected RSSI at the actual timestamp from node n of a total of N nodes. θ and ν are the current estimated orientation and velocity respectively.

In order to create these vectors, it is necessary get the current orientation and velocity. Furthermore, the initial position to start the tracking should be also estimated. All these parameters can be obtained using the previous proposed system to get the velocity, orientation and position before tracking. RSSI is collected for each node and synchronized in a time window of 100 ms. If from one or more nodes no measurement is collected inside the window the RSSI value is set to $-99dBm$.

Before enter in the network the data is normalized to be in the range 0-1. Each parameter has his own limits. To normalize the position the size of the room is used. In order to get the RSSI in the range 0-1 the limits were set to -35 dBm (higher RSSI value possible) and -99 dBm (lower RSSI value possible). Finally orientation is in the range 0-360 and can be directly normalized, in contrast velocity was normalized taking as superior limit $2.5 \frac{m}{s}$.

The dataset is composed by a set of 5000 tracks generated by data augmentation of 300 labeled tracks collected at different day hours. The data augmentation consists in this case in flip routes, cut parts and merge tracks. The period of each track collected is 30 seconds.

Two Recurrent Neural Network have been tested, One to one RNN and Many to Many RNN. The first proposal consists in only sequences of length equal to one or in other words the input is directly the feature vector. The second idea is to use a many to many configuration where the input is a sequence of feature vectors. The first configuration map directly the feature vector to a position. The second proposal takes as input the sequence and for each iteration

Figure 5.28: Recurrent Neural Network system architecture. It is composed by the WSN that collect RSSI measurements and the velocity and orientation estimators. The recurrent neural network predicts next estimated position using the proposed feature vector. Tracking recovery is applied when no measurements are received and use a Conditional GAN that generates RSSI maps and a second Recurrent Neural Network with a different input.

outputs an estimated position that is used as input in the next feature vector of the sequence.

The complete system architecture described is drawn in Figure 5.28. The recurrent neural network configuration consists of 4 cells of one LSTM layer and a final neurons layer with two neurons that regress the position values (x and y) to the real ones (from the dataset) using the mean squared error loss function. Adam optimizer was used with a learning rate of 0.001 and a batch size of 128. A modification of the mean squared error loss function is also proposed in order to filter and smooth noisy estimations in time (Equation 5.48) for the many to many configuration:

$$Loss = \sum_{t=1}^T \|\mathbf{P}_t - \mathbf{P}_t^{gt}\|_2 + \lambda \|\mathbf{v}_t - \mathbf{v}_{t-1}\|_2 \quad (5.48)$$

Where $\mathbf{P}_f$ and $\mathbf{P}_f^{gt}$ represent the estimated position (x, y) from the network and the labeled position in the dataset (ground truth) respectively. T is the total number of steps in a complete sequence of data. A regularization parameter is included and use the velocity information from the actual position and the previous position. Hyperparameter λ control the degree of the smooth, higher values produce more smoothed trajectories.

5.3.5.2 Results using RNN tracking

The proposed architecture was tested using XBee technology in the scenario presented in Figure 5.8 where 4 nodes are allocated at the top of the corners of the room.

To check the accuracy of the proposed network in different configurations 100 additional tracks of 30 seconds have been collected and labeled and data augmentation was performed to have a total of 2000 routes with different lengths (between 100 and 300) to compare the estimations of the RNN with a new validation set different from the used for training and testing. The proposed configurations of the main RNN was a One to One (OtO) RNN and three Many to Many (MtM) RNNs with different sequence lengths of 5, 15 and 30 steps respectively.

Larger lengths of the sequence incur in more errors predicting new values. All the networks have been trained for 30 epochs and all the Many to Many configuration was trained using both MSE loss function and the proposed MSE function with the regularization parameter with lambda settled to 10 (value experimentally obtained after several tests). Table 5.5 collects

the mean and deviation of the error using different RNN configurations after training phase.

Table 5.5: Mean Error and Deviation for different RNN configurations (FV1)

OtO = One to One RNN, MtM = Many to Many RNN		
RNN configuration	Mean Error (cm)	Error Deviation (cm)
OtO MSE	138.54	80.45
MtM MSE (length 5)	77.87	61.34
MtM MSE (length 15)	65.67	41.98
MtM MSE (length 30)	71.48	58.12
MtM MSE+Reg (length 5)	75.23	56.87
MtM MSE+Reg (length 15)	63.76	38.56
MtM MSE+Reg (length 30)	69.02	49.91
Swiblux (Previous work [94])	110.67	95.43

The results are promising showing a reduced mean error and a low deviation. One to One RNN achieved the worst results. This is probably due to no temporal information is added in only one timestep and the higher changes between parameters in time (majority due to orientation and velocity) are not sufficient information to get better results. In contrast, Many to Many configurations show better results with a low variance in the error. Temporal information is useful for these type of networks in order to learn efficiently from the input data. The proposed RNN with input sequence length of 15 presents the best results during the experiments. As was commented, more data in the sequence can be not ideal to estimate very far values in time. It can be shown that sequences of 30 timesteps are worst and the deviation increase to a value very near to the presented for the RNN with 5 timesteps.

Furthermore, there exist an interesting improvement using the proposed MSE Loss function with a regularization parameter decreasing both, mean error and deviation. It can be argued that the regularization parameter is effective in order to smooth the trajectory filtering important outliers to a nearest real value giving a increase in accuracy of a 3% in both error mean and deviation. Figure 5.29 presents the Cumulative Distribution Function (CFD) of the errors for the validation trajectories.

In addition, in order to compare the results with a similar work, our previous proposal have been used with this new dataset. The results showed that our previous proposal achieve worst performance in the estimation accuracy. The temporal learning is an important difference between both architectures and this work is not able to adapt it better than the new proposal

to the scenario changes in time.

Figure 5.29: CDF error tracking in the routes validation set for different RNN configurations and the proposed modified loss function with a regularization parameter.

Finally experiments with different trajectory periods have been performed. Results are drawn in Figure 5.30.

It is important note in this experiment that exists and increment in the mean error after large periods of time. For example, after 30 seconds the first RNN error increase to a value of around 120 cm where at the beginning this error was around 65-70 cm and the same happens with the second RNN but with a more important increment after larger periods. The behavior showed in this plot is due to the errors along time in the estimations. The first RNN is able to filter better the estimations along time thanks to the more complete feature vector used. The second RNN increase the error quickly after time due to only RSSI and position information (with previous error) is given.

To solve or decrease the degree of importance of this problem for the first RNN can be interesting start again the tracking process estimating the first position every fixed period of time. For the second RNN, a fixed period can be also selected as effective period for tracking recovery, if after this time no measurements are collected it can be considered that the tracker is completely loss and need start over again the tracking process.

Figure 5.30: Error increment after different periods with the system working for both RNN with feature vector (FV1) and sequence length 15 and RNN with feature vector (FV2) and sequence length 15.

Finally, most important results presented was a Fingerprinting dataset collected along several days using also the device equipped with the Inertial Measurement Unit to have all the parameters involved in the proposed feature vector as input of the Recurrent Neural Network. Different RNN configurations have been tested and the proposed MSE loss function (Equation 5.48) increase the accuracy in the position estimation in relation with other combinations up to 3%. The mean error and deviation are very interesting results with high accuracy.

5.3.6 Tracking recovery module

During person tracking measurement losses can happen in short periods and can cause tracking losses due to no information is received. To prevent these situations a Generative Adversarial Network is applied to generate RSSI maps to get measurements when they are not available. Using the RSSI predicted measurements a new position can be estimated using the following

feature vector (Equation 5.49):

$$FV2 = [x_{t-1}, y_{t-1}, RSSI_1, RSSI_2, \dots, RSSI_N] \quad (5.49)$$

Where x_{t-1} and y_{t-1} are the previous estimated position and $RSSI_n$ is the generated RSSI for each node in the previous estimated position where N is the total of nodes.

Figure 5.31: Conditional Generative Adversarial Network architecture. The input is a vector of Gaussian noise and the proposed one hot labels. The output of the Generator part after training is the RSSI maps. The Discriminator part help to learn which images are Fake or Real. Numbers represents in case of convolutions the number of filters employed and, in the neurons layers, the number of neurons.

A second Recurrent Neural Network was used to learn to predict next position in a predefined window time where the input data is a feature vector with only the previous positions and the estimated RSSI extracted from the generated RSSI maps by the GAN model. In the new

feature vector, x_{t-1} and y_{t-1} are the previous estimated position before loss the tracker or the estimated new position with the second recurrent neural network in the next iterations. Both previous configurations for this RNN have been tested.

A conditional GAN has been used to generate feature maps that depends on time. Based on the previous examples of collected RSSI along different hours of the day can be argued that RSSI change significantly to take in account this variation in the RSSI synthetic generation. To accomplish this task a new dataset has been created with RSSI maps and the GAN model have learn to predict these feature maps in time using conditional inputs.

The proposed conditional GAN model is presented in Figure 5.31. The input of the generator part is a fixed size Gaussian random noise of 100 values concatenated with the label vector. This vector is composed by a 48+N values where the first 48 values represents a split of the day RSSI measurements in blocks of 30 minutes. The last N values represent which node is active from the available. Then finally, the input vector contains the random Gaussian values and the labels where the concrete time period and the selected node have a 1 in the label and the other contains zero values. As an example, if a RSSI map from node 1 (from a total of 5) at 1:30 AM want to be generated this vector is as follow (Equation 5.50):

$$labels_{GAN} = [0_1, 0_2, 1_3, 0_4, 0_5, ..., 1_{49}, 0_{50}, 0_{51}, 0_{52}, 0_{52}] \quad (5.50)$$

The proposed model generates RSSI maps learned from the created database along time. To create this dataset a measurement synchronization from each cell at the same time was performed in window of 10 seconds for each node. The RSSI rounded mean has been taken as the candidate to create the feature map inside the predefined windows time at each cell. An interpolation between cells have been used to connect all the cells and to create the final RSSI map image.

The color code has been predefined to map directly a RSSI value from a concrete pixel that represents a concrete position in the proposed scenario. An example of the created RSSI maps is presented in the top part of Figure 5.32 where the color bar represents the RSSI intensity in the predefined RSSI range (-35 dBm to -99 dBm). The size of the generated images is 360x200 pixels and during training will be reshaped to fix the CGAN model input to a size of 128x128 pixels. Finally, the dataset is composed by 48 folder (representing every thirty minutes in a day) and 4 subfolders in each one with the maps generated for each node.

Figure 5.32: Example of original created RSSI maps in the proposed room. *Top:* Real RSSI maps in different times of the day. *Bottom:* Generated RSSI maps using the proposed Conditional GAN in different day hours.

In Figure 5.31 the Convolutional and Transpose Convolutional blocks are formed by a 2D Convolution or 2D Transpose Convolution respectively, Batch Normalization with momentum equal to 0.8 and LeakyRelu activation with alpha 0.2. Furthermore, the figure contains the number of filters employed in the convolutions and the number of neurons in these layers. Finally, the first neuron layer in the generator is activated by a ReLu function and the last layer of the discriminator use a Sigmoid function to decide between real or fake.

Next experiment shows the performance of the generated RSSI maps by the proposed Conditional GAN along time. As an example, bottom part of Figure 5.32 presents examples of the reshaped generated maps to the original size in different time hours. In order to check the reliability of these maps the mean RSSI map have been obtained for each time hour in the day and node as the mean of the converted pixel images to RSSI values for both real images and 1000 generated images at each time. After this operation a total of 192 (48 time intervals * 4 nodes in each interval) arrays of size 320x200 (RSSI map size) is obtained for both real and generated images. The error between these arrays is calculated as follows:

$$E_{map} = \sqrt{\frac{1}{WH} \sum_{w=1}^W \sum_{h=1}^H \left(\frac{1}{T} \sum_{t=1}^T (\mathbf{RSSI}_{Real,t} - \mathbf{RSSI}_{Gen,t})^2 \right)} \quad (5.51)$$

Table 5.6: Mean Error and Deviation for different configurations (FV2)

OtO = One to One RNN, MtM = Many to Many RNN		
RNN configuration	Mean Error (cm)	Error Deviation (cm)
MtM MSE+Reg (length 5)	128.04	98.67
MtM MSE+Reg (length 15)	97.51	78.06
MtM MSE+Reg (length 30)	116.87	85.38

Where W and H are the width and height of the image respectively, T represented the total number of arrays for comparison and $\mathbf{RSSI}_{Real,t}$, $\mathbf{RSSI}_{Gen,t}$ are the generated arrays for the time period t for the real and generated images.

5.3.6.1 Results with proposed CGAN for RSSI Map generation

Results show that the error between maps with the proposed metric (Equation 5.51) is 2.69 dBm and deviation for real and generated images is 4.19 dBm and 5.13 dbm respectively. These results show the effectiveness of the generated maps in order to generate new RSSI measurements in time. In the majority of the cases the error in RSSI estimation in one point is around 2.5 dBm that is a very good estimation for the majority of cases in the dataset. Deviation for both groups of images is also very similar achieving generate maps with an adequate range of RSSI values.

Using the proposed CGAN and the generated maps new RSSI measurements can be extracted to estimated tracked positions when the nodes don't receive measurements from the personal device in short time periods.

In order to test the performance of the tracking recovering module with the generated RSSIs a second Recurrent Neural Network similar to the proposed in Section 5.3.5.1 was used to learn to predict next position in a predefined window time where the input data is a feature vector with only the previous positions and the estimated RSSI extracted from the generated RSSI maps by the GAN model. The network is trained using the proposed feature vector (FV2). In this case only the Many to Many configuration with the MSE loss function with regularizer is used.

Results (Table 5.6) are again promising having a reduced error in the estimations and achieving the configuration with sequence length of 15 the best performance. It can be shown that the error is higher in comparison with the previous RNN. This is due to two main factors, firstly, less parameters are available in this feature vector and orientation and velocity are important parameters which work properly in sequence predictions. Secondly, the RSSI values are not real and are the generated by the network. Depends on concrete factors at each moment these estimations can be not good enough producing very high estimation error and losing the real tracker completely.

Finally, summarizing the results, the Conditional GAN have been tested comparing the mean and variance of the RSSI maps with respect to the real RSSI maps generated showing that the similarity between the behavior of the RSSI in the real maps is very similar to the presented in the generated maps when lots of maps are used to get these values. Additionally, the proposed recurrent neural network is able to learn efficiently from the input feature vectors learning to predict the position using the generated maps when no measurements are presented having good accuracy in the estimations in short window times.

5.4 Research contributions for indoor tracking with wireless and inertial sensors

Several experiments have been performed exploiting different ways to model wireless signals behavior in indoor environments. Firstly probabilistic and fusion approaches have been presented and are supported by the paper [68].

Secondly, experiments combining robust tracking algorithms like particle filter in combination with inertial sensors and deep neural networks to get an accurate estimated position are explained in [94].

Finally, a sequence model using recurrent neural networks to work in time with the combination of GANs to predict values when losses appears is presented in [128].

Chapter 6

Proposed image based IPS solutions

6.1 State of the art IPS solutions without learning algorithms using Image devices

6.1.1 Foreground-Background Segmentation in ISS

Foreground-background segmentation is the first important step for ISS. The goal is to separate the object or moving object (foreground) and the environment (background). It commonly referred also as background modeling, background subtraction, or change detection. Many foreground-background segmentation techniques have been proposed, especially for visible/video surveillance. Several review paper also available that focused discuss foreground/background segmentation methods. Some of the most known methods are generalized mixture of Gaussians (MOG), non-parametric kernel density estimation, and codebook [129]. Examples are shown in Figure 6.1.

Obviously, any possible techniques for foreground/background segmentation depend on the corresponding sensor modality. In [130] the most common techniques in this field have been collected. Most of the proposed method in background-foreground segmentations employed only single sensor modality, and particularly using visible camera. Obviously, combining different sensor modality would make the system more robust or simplify the processing process for segmentation. For example, by combining visible camera and range data the

background-foreground segmentation task become easier. Changes in the background can be easily filtered out by excluding all data outside of the observed area in the range data and the visible/image data is used for fine segmentation.

Figure 6.1: Example of two simple algorithms for background segmentation in pedestrian and pedestrian and vehicles environments. Left: Mixture of Gaussians (MoG algorithm). Right: Median Blur algorithm.

6.1.1.1 Object Detection and Classification in ISS (before Deep Learning solutions)

The ability to automatically detect and classify objects (such as human and vehicle) is one of key component in ISS. For a machine or computer, detecting object like human is a hard job due to wide range of possible appearance as result of changing articulated pose, clothing, lighting and background. Huge methods have been proposed for people detection based on visual camera. Object detection is an extensive field of study nowadays. Several approaches exist using image processing and computer vision techniques. The combination of these algorithms with machine learning techniques gave interesting results and moderate precision in the detection task.

The most common technique [131] is train a machine learning algorithm such as Support Vector Machine (SVM), Random Forest or Gradient Boosting between others to learn about the

objects in an image extracting features using algorithms as Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), etc... These detectors are created in combination with a sliding window and image pyramid algorithms to cover all the image in different scales and obtain a detector that is invariant in translation and scale. Depends on the feature extractor the detector can be also invariant to rotations.

In a more recent benchmarking effort, FPDW [132] has the best overall performance, but If computational cost is not a consideration, then MULTIFTR+MOTION [133] is the best choice. But all these methods are so old where the last work was presented in 2013 before the statement of Deep Learning solutions that are today the state-of-the-art methods in object detection and classification.

The main problem with this approach is, which is the best feature extractor for each problem? Which machine learning algorithm should i use? How to tune all the parameters involved in the algorithms to work in most of the use cases? Other related problem of this methods are very slow computational performance if the image size is high, the sliding window is little and a lot of pyramid scales are used.

Figure 6.2: Most extended person detection chain algorithm using HOG and SVMs. The first step use background segmentation algorithms to extract possible object in movements. HOG descriptor and SVM classifier is applied over the detection to get if is a person or not.

In [134] a system is proposed for people detection using a bottom-up top-down detector, based on lidar data. The bottom-up detector learns a layered person model from a bank of specialized classifiers for different height levels of people that collectively vote into a continuous space. In the top-down step, the candidates are classified using features that are computed in voxels of a boosted volume tessellation.

While in X Banedek et al map the 3D lidar point data into depth image, and performing

people detection in 2D. In [135] a people detection approach based on RGB-Depth sensors that provide both image and range data is presented. Most of the proposed methods in objects detection and classification focusing only for couple of object types for example humans and cars. In fact, in the real setting, there are a lot of object that should be considered also for example different type of animal or other subject that have potential threat for security or safety.

6.1.2 Object Tracking and Re-Identification in ISS

After object detection, surveillance systems generally track the object in the spatiotemporal domain. Object tracking in a realistic scene is a challenging problem due to illumination changes, occlusion, clutter, sensor motion, and other issues. A large number of visual tracking algorithms (based on visible camera) have been proposed in last years.

Several trackers apart Kalman Filters [136] and Particles Filters [137] from scratch exists but need a previous stage of detection or enter manually the initial bounding box objects in the image to start the tracking. Object tracking methods based on visual camera can be classified into five groups: model-based, appearance-based, contour and mesh based, feature-based, and hybrid methods. Several review papers that focused on visual tracking problem are available such as [138, 139, 140].

All these algorithms learn directly from the proposed region of interest to track. BOOSTING Tracker [141] is based on used to power the machine learning behind Haar cascades (AdaBoost), but like Haar cascades, is over a decade old. This tracker is slow and doesn't work very well. MIL Tracker [142] present better accuracy than BOOSTING tracker but does a poor job of reporting failure. Kernelized Correlation Filters Tracker [143] is faster than BOOSTING and MIL but not handle full occlusion well. CSRT Tracker [144] (Discriminative Correlation Filter with Channel and Spatial Reliability) tends to be more accurate than KCF but slightly slower. MedianFlow Tracker [145] does a nice job reporting failures, however, if there is too large of a jump in motion, such as fast-moving objects, or objects that change quickly in their appearance, the model will fail. TLD Tracker [146] explicitly decomposes the long-term tracking task into tracking, learning and detection. MOSSE Tracker [147] is very fast, not as accurate as CSRT or KCF but a good choice if you need pure speed. Some performance comparisons between these algorithms are presented in Figure 6.3.

Figure 6.3: Left: Box plot illustrating averages and distributions of frame rates. Right: Success plots of the five tested trackers showing the fraction of frames successfully tracked as a function of required Jaccard index. Results presented in [7].

Recently, some attempts have been done for people tracking using other than visible camera, such as using radar, lidar etc. For example, in [148] authors use stereo range data for real-time multi-person tracking. They did not only analyze 2D image, but also the range information from stereo camera. In [149] a development of an automatic target classifier (such pedestrian and vehicles) using ground surveillance radar is presented. Based on lidar data, authors in [150, 151] proposed 3D people tracking using multi-target multi-hypothesis tracking approach. For people detection they employed a bottom-up top-down detector. In [152] an approach on real-time 3D people surveillance, with probabilistic foreground modeling, multiple persons tracking and on-line re-identification is shown. The tracker module was also tested in real outdoor scenarios, with multiple occlusions and several re-appearing people during the observation period.

6.1.3 Behavioral Analysis

There is an increasing interest to automatically analyze surveillance scene not only in the "object level" (such as detecting, tracking), but also further into "event level". Particular interests such are automated human behavior analysis [153, 154], group behavior analysis [155], crowd analysis [156, 157], and event analysis. Some review papers have been devoted to this topic [158, 159].

Human behavior analysis can play a significant role in security by decreasing the time taken to thwart unwanted events and picking them up during the suspicion stage itself. Analysis of human behavior although crucial, is highly challenging. Basic component in human behavior analysis is classifying the human behavior. Different ways have been proposed to classify human behavior. Authors in [160] proposed simple classification: normal and abnormal. In [161] authors expand the classification into normal, unusual and abnormal. Previously, in [162] authors classified the activities as positive, neutral and negative activities.

Human can be monitored as isolated individuals, groups of people, or crowds. Examples of group events are people fighting, people being followed, people walking together, terrorists launching attacks in groups, etc [155]. In [156] is proposed a method for identifying five crowd behaviors (bottlenecks, fountainheads, lanes, arches, and blocking) in visual scenes. In [163] is proposed an activity monitoring framework for recognizing behaviors, involving either isolated individuals, groups of people, or crowds, in the context of visual monitoring of metro scenes, using multiple cameras. The most common pipeline for these algorithms is shown in Figure 6.4.

Figure 6.4: Detection and people tracking is performed to get information about person movement in time. Next steps presents feature extraction like "Key-points". Finally, using this information the estimated behavior is obtained.

6.2 State of the art IPS solutions using learning algorithms and Image devices

6.2.1 Machine/Deep Learning for person detection and tracking using image devices

Deep Learning approaches are the new novel field to perform these tasks and improve the detection and tracking results [164]. Using convolutional neural networks in combination with computer vision algorithms the detection part works better for a large number of objects in the learning phase in comparison with machine learning methods. The problem still remains in the use of a sliding window and pyramid image to detect parts of the image that can be potentially and interesting object to track.

The fast growth of Deep Learning proposed several algorithms to perform object detection in image in a pure Deep Learning manner (Figure 6.5). Several algorithms have been presented starting with RCNN (Region Based Convolutional Neural Network) family (Figure 6.6). These family incorporate a region proposal part and a classification and regression learning task to classify the objects and fits the better bounding box to the object. RCNN [165] evolved to Fast-RCNN [166] and this last evolve to Faster-RCNN [167] that accomplish all object detection task in a complete Deep Learning framework.

Figure 6.5: Deep Learning object detector algorithms in time. In red the most known algorithms are marked.

Other algorithms appear at the same time such as Single Shot Detector (SSD) [168] and the latest modification called Receptive Field Block Net (RFBNet) [169] and You Only Look Once (YOLO) in three different versions [170, 171, 172]. These two approaches changed the concept of use a region proposal as an external part and combine all the detection and classification tasks in only one network that propose regions during the phases of the algorithm.

Others concepts appears like Region Based Fully Convolutional Neural Network (RFCN) [173] that is an entire convolutional approach to perform object detection. Feature Pyramid Network (FPN) [174] was a proposal to include scale invariant to the networks. Local Binary Convolutional Neural Networks (LB-CNN) [175] have been a first approach to modify convolutional filters in order to take advantage of other parameters learned in the filters to detect better other features such as textures or increase the speed of the algorithm working with tiny datasets.

Figure 6.6: Region Convolutional Neural Network (RCNN) family.

Mask-RCNN [176] extend the Faster-RCNN with a new branch that perform object segmentation in conjunction with the other branch that classify the object and estimate the bounding box of the object. One of the most recent works is RetinaNet [177] that is a new simple network that use FPN and a new Loss Function that increase the accuracy and the speed in training phase and gives better results in testing phase.

Other networks have been created concentrating the work in the keypoints detection over the human body (Figure 6.7). OpenPose [178] proposed a series of algorithms to draw a skeleton based on several keypoints detected and also perform hand and head keypoints detection.

Latest versions of this algorithm provide also tracking between skeletons in different frames. In the other side other works like DensePose [179] present a complete framework with a very complex annotated dataset to collocate a 3D surface over the body persons in a 2D image. No tracking is provided for this algorithm. Alpha Pose [180] is an accurate multi-person pose estimator, which is the first real-time open-source system that achieves 70+ mAP on COCO dataset [181] and 80+ mAP (82.1 mAP) on MPII dataset [182]. To match poses that correspond to the same person across frames, it is also provided an efficient online pose tracker called PoseFlow [183].

Figure 6.7: Keypoints human detection algorithms. Left: OpenPose (Skeleton, hands and face detection). Center: AlphaPose (Real time detection in crowd areas). Right: DensePose (Estimated 3D mesh).

In contrast more, recent algorithms are focus on tracking using only Deep Learning algorithms. Siamese Fully Connected Networks (SiamFC) citeBertinetto2016FullyConvolutionalSN, Re3 [184] and GOTURN [185] are recent works that perform tracking over a given object in the image. Then, no detection is performed, the object must be provided previously manually or using any object detection algorithm. The combination of the image with other sensor information can provide more accurate results in the detection and tracking tasks. Fusion algorithms can be used to take information from multiple sources (Cameras, Lidar, GPS, Accelerometers, Gyroscopes, Magnetometers, Depth...) and merge the data provided to perform several tasks at the same time and fuse the results at the last part of the framework (Late Fusion) [186, 187] or combine the information in the first steps to provide a unique result (Early Fusion) [188, 189].

But there also exists a huge effort in order to create architectures that perform both, detection and tracking in a unique network architecture. As the environment evolves, new deep learning solutions are gained the attention within tracking-by-detection paradigm, consider-

ing architectures that combines the power of CNN object detectors and the ability of RNN to predict future states in time. Some new approaches as ROLO (recurrent YOLO) [190], Mf-SSD [191] and D&T (Detect & Track) [192] are very interesting ideas to compare with our scheme. Finally a new version of D&T [193] also estimates skeleton keypoints in the human body.

In the recent years, the proposed methods have started to be applied over panoramic images to solves problems associated to object detection, tracking, depth estimation and saliency detection. In [194] Yolo object detector is applied over different projected points of view selected in the image. Detected objects are reprojected to the panoramic image and bounding boxes are refined. The work presented in [195] apply region based convolutional neural networks to perform object detection directly on panoramic images. Tracking algorithms have been also tested in panoramic images [196, 197].

Authors in [198] present the creation of a novel dataset based on 3D scenes to perform depth estimation over panoramic images. Finally [199] present a dataset for Saliency Detection in and [200] provide an architecture to learn to predict these maps. But some problems are still presented, in one hand, the lack of a huge variety of annotated datasets for 360° images makes difficult train deep neural networks to solve efficiently these tasks in all possible domains. By the other hand, panoramic images present important inherit distortions patterns that make difficult work directly over the image without change to different NFoV projections.

Figure 6.8: Deep 360 Piloting network architecture. Object detection and Recurrent operator to learn in time to track more salient actions.

Active work and research are been doing to deal with distortions using spherical projections to work directly in the spherical domain. Several works proposed a new interesting type of

deep learning layer known as Spherical Convolution [201, 202, 203, 204, 205] and takes the advantage of project the convolutional kernels into the sphere. Additionally Spherical Max Pooling have been exploited to be applied also in a spherical way [206]. Authors in [202, 206] apply Spherical Convolutions to perform object detection achieving better accuracy in NFoV images projected into the panoramic image.

All these ideas have been combined to improve results in auto cinematography creation using panoramic images. First work in this field is known as Autocam [207]. To avoid problems with distortions the framework proposed used NFoV images projected from the panoramic image and introduced an algorithm to get the final scanpath. Other work applied object detection in combination with recurrent neural networks [208] and learn a policy function to track the main object automatically (Figure 6.8). Finally several authors proposed the use of saliency detection [209, 210, 211, 212] due to some 360° datasets for saliency map prediction are available and combine other extracted features to train a network that predict the final path from the saliency maps. All these works are the state of the art in automatic cinematography from panoramic images but don't present any solution to solve or decrease problems related to distortions due to work directly with this type of image.

6.2.2 Machine/Deep Learning for person re-identification using image devices

Actually, a huge number of cameras cover every corner and the data generated from digital surveillance has become so massive that it is impossible for human operators to make sense out of. The rise of machine/deep learning vision algorithms and the advances in hardware that can handle parallel processing (GPUs) made possible the application of more complex processes to discover hidden patterns in data.

The vision technique applied to solve re-identification problems is slightly different from other vision tasks such as object recognition/detection or segmentation. One formal definition can be the following:

“Given an image/video of a person taken from one camera, re-identification is the process of identifying the person from images/videos taken from a different camera with non-overlapping fields of views. Re-identification is indispensable in establishing consistent labeling across multiple cameras or even within the same camera to re-establish disconnected or lost tracks.”

It is necessary find a matching person (in a gallery of images) for a given image using not (just) the facial features but features found in his/her entire body (like clothing, height etc). In other words, assigning a stable ID to a person in multi-camera setting. Usually the re-identification is constrained to a small time period and a small area covered by cameras. Humans are easily able to Re-ID others by leveraging descriptors based on the person's face, height, and build, clothing, hair style, walking pattern, etc but this seemingly easy problem is extremely difficult for a machine to solve.

The main challenge comes from the intra-class variations, i.e. the variations in appearance of the same person across different cameras. Prior to Re-ID the system has to detect a person and define the bounding box of the person in an image. Human body is very deformable then is challenging detect it in a proper way in all possible situations.

In a video input we need to be able to establish correspondence between detected subjects across frames. This process is called tracking. Tracking multiple persons is also a challenging task. Intensity of daylight, shade, reflected light from colored surfaces, indoor lighting can cause the same subject to appear in different shades and colors across cameras doing hard the detection and tracking processes. Furthermore, in crowded environments partial or even complete occlusion of persons by others presents challenge in extracting features.

Uniform clothing at schools and even some workplaces will confuse Re-ID algorithms which extract information from clothing/appearance. When the input is a video, we can even think about gait analysis etc for Re-ID.

There are a set of established methods in the literature that have been employed for these tasks with successful results and promising accuracy results in several different datasets.

Contextual methods are where we use external information such as position, angle etc. of cameras to aid Re-ID. By leveraging the information about location of cameras and the time of capture of images we can reduce the gallery size for a query. Also, by knowing entry/exit locations, velocities, transition times between cameras we can continuously track a person. If there is overlapping between fields of views of cameras and no blind spots then we can track a person across cameras without needing much Re-ID.

Non-contextual methods can be divided into two: active and passive. Passive methods do not use any machine learning for descriptor extraction and matching whereas active methods do. Active methods are further subdivided into 3 main categories:

- Color calibration: modelling the changes in the appearance of objects between 2 cameras by learning a brightness transfer function. Re-ID is done on the transformed appearance models.
- Descriptor learning: in this method we attempt to learn the most discriminative features or a discriminative weighting scheme for multiple features for Re-ID.
- Distance metric learning: this method focuses on learning appropriate distance metrics that can maximize the matching accuracy regardless of the choice of appearance representation.

Figure 6.9: Most extended Re-Identification methods.

Deep Learning methods are now breaking the barriers of older methods trying to learn directly during the training process the final feature vector used by the matching algorithm. The results are very promising showing that is possible extract a robust feature vector when huge datasets are available to learn from persons in different cameras, positions and lighting conditions automatically without defining your own metrics and giving all the complexity in the neural network deployment.

Most of the Deep Learning re-identification approaches take the advantage of CNNs [213, 214, 215, 216, 217, 218, 219] in order to have architectures with object categorization purposes. Some modifications inside architectures are introduced in order to reflect that images datasets in this field only contains persons (only one category) where the majority is upright. In this pose is easier extract some useful information from the persons [220, 221, 222, 223] and it is

possible to add some attention layers at the middle or the end of the network. [224, 225, 226] presents mechanism to put the attention in the background in order to reduce the effect of very different landscapes.

In [227, 228, 229, 230, 231], parts of the body are extracted and learned using body-pose detectors. In [232, 233, 234], different CNN branches perform local or global attention in the image. Works [235, 236, 237, 238] extract multi-level features at different layers to be combined at the end.

[239] proposed a feature fusion deep neural network to regularize CNN features jointly by handcrafted features (best performance obtained from ELF descriptor [240]). In [241], low-level descriptors including SIFT and color histograms are extracted from the LAB color space over a set of 14 overlapping patches. PCA and embedded using Fisher vector encoding produce the final feature representations. Xiao et al. [242] proposed learning deep features representations from multiple datasets by using CNNs to discover effective neurons for each training data set. for each data set, they performed the forward pass on all its samples and compute for each neuron its average impact on the objective function.

In [243], a Siamese neural network has been constructed to learn pairwise similarity. The part pairs are matched through three independent Siamese networks, and finally are fused at the score. Wang et al. [244] designed an ensemble of multi-scale and multi-part with CNNs to learn image representations and similarity measure jointly. Wang et al. [245] developed a CNN model in order to jointly learn single-image representation (SIR) and cross-image representation (CIR) for person re-identification task. Liu et al. [246] utilized a deep learning model to integrate a soft attention based model in a Siamese network. The model focuses on the important local parts of input images on pair-based Siamese model.

Chen et al. [247] proposed a deep ranking framework to jointly learn joint representation and similarities for comparing pairwise images. Franco et al. [248] proposed a coarse-to-fine approach to achieve a generic-to-specific knowledge through a transfer learning. Later, the same authors proposed a novel type of features based on convolutional covariance descriptor in [249]. Wang et al. [250] proposed a pairwise Siamese model by embedding a metric learning method at the top of the network to learn spatio-temporal features. In [251], a Siamese network takes a CNN learning feature pair, and outputs the similarity value between them by applying the cosine/Euclidean distance function. The multi-task deep learning proposed in [252] aimed to use a separate soft- max for each auxiliary task as identification, pose labeling, and each attribute labeling task.

A Strict Pyramidal Deep Metric Learning approach is proposed in [253], in which the Siamese network composed by two strict pyramidal CNN blocks with shared parameters between each, and produced salient features of an individual as the output of the network. A novel scheme, called deep multi-view feature learning (DMVFL), proposed in [254] aimed to combine handcrafted features with generated deep features using by CNN model and embedding a metric distance method at top of the network in order to learn metric distance. A two channel convolutional neural network by new component named Pyramid Person Matching Network (PPMN) proposed in [255] with the same architecture of GoogLeNet; in which the network takes a pair of images as input, and extracts semantic features by convolutional layers; and finally the Pyramid Matching Module aimed to learn corresponding similarity between semantic features based on multi-scale convolutional layers. Qian et al. [256] proposed a multi-scale model which is able to learn discriminant feature from multi-scales in terms of different levels of resolution. A saliency-based learning strategy is adopted to learn weight important scales.

6.3 Proposed Cameras Architecture

Figure 6.10: Proposed Cameras Architecture for person detection, tracking and reidentification.

First step consists of person detection algorithm to get the bounding box used to cut the person in the image and extract re-identification features. After detecting persons, a tracking algorithm is applied to ensure that the same person is detected over the video with the same ID. A tracking recover approach is used in order to maintain the stability of the system when the tracking algorithm fails due to complex occlusions or other conditions in the image.

Several cameras will work in parallel extracting features frame by frame for all the detected persons. All this information will be collected from each camera in a common algorithm that will perform the matching process to get the equivalences between extracted feature vectors in time. Object detection part used deep learning approaches in order to perform detections over a large number of objects. There exists some pretrained networks that can work in real time using a powerful GPU. Some of these algorithms are YOLO [170, 171, 172], Mask-RCNN [176], SSD [168], Retinanet [177], etc... that are ones of the state-of-the-art object detection algorithms in deep learning. Our framework uses these algorithms trained on COCO dataset to be retrained in VOC-PASCAL dataset [257] only using annotated persons classes.

After detection tracking is applied to get new IDs for new persons in the image and to associate previous IDs to existing persons that are under tracking. Several networks have been used to get the re-identification features from the standard ResNet [258], VGG [259], SqueezeNet [260], Inception [261] and MobileNet [262] architectures trained to perform classification over reidentification datasets. Some other more complex network concentrated in the reidentification tasks have been also trained and use it to get features as MuDeep [256], HACNN [224], PCB [220] and MLFN [235].

Finally, a feature vector is extracted and sent to the main processing algorithm that will collect all the information provided by all the cameras. A synchronization process take place in order to filter situations that is impossible that happens in several cameras at the same time. An optimization problem is proposed to solve the data matching using a distance metric and a final threshold to decide if is a match or not.

Normal field of view (NFoV) cameras are the first type of cameras proposed. The selected camera was Vivotek FD8379-HV [263] which main features are 30 FPS at 2688x1520 resolution, 4 MP and 114° Horizontal FoV to cover large areas with a single camera. The second type was fisheye cameras and the one selected in this work was the Vivotek FE8180 [264] with 5 MP and 15 FPS at 1080p Full HD in fisheye mode. Finally, one 360° cameras is also included in this work and the model selected was the Ricoh Theta V [265] with 3840×1920 resolution (4K) at 30 FPS. An example of the image collected by each camera is presented in Figure 6.11

Figure 6.11: Example of the images collected by each type of the proposed cameras. *Top left:* NFoV camera. *Top Right:* Fisheye camera. *Bottom:* 360° camera.

6.4 Image projections

Before apply detection and tracking algorithms directly over the images some considerations should be analyzed and taken in account. In the NFoV camera normal images are collected and can be used directly as input of the detection algorithms. Due to the high FoV of this camera, some distortions appear far from the center near to the borders. These distortions are not so important but this problem can be corrected getting the undistorted images. Undistortions will be presented in a different problem in Section 6.5. In contrast, in fisheye and 360° cameras some problems appear by the inherent distortions of the collected images directly from the

cameras to present all the image information in an rectangular view.

Regarding fisheye camera, the distorted image can produce detection errors due to the point of view is from the ceiling to the floor. Persons appear in different sizes and perspectives if they are in the perpendicular point under the camera (only head and shoulders) or are far from the camera where the complete person (from feet to the head) is presented. In order to check the detection algorithms the fisheye image and the unwrapped image have been tested. To obtain the unwrapped/unrolled image from the fisheye view two steps are required. First step comprises select the indoor and outdoor radius to unroll the image. Figure 6.12 shows an schematic example of the process:

Figure 6.12: Fisheye unrolling process to map points in the input image to a pseudo panoramic view.

This process takes as input the fisheye image and define the internal and external circle around the image will be unrolled. Is important to note that depending on the selected radius for the internal circle all the information contained in this circle will be lost. The with these definitions and knowing that the center of the circles is in the center of the image represented as Cx, Cy it is possible define the final width and height of the pseudo panoramic unrolled

image.

$$\begin{aligned}
 Ws &= width \\
 Hs &= height \\
 Hd &= R2 - R1 \\
 Wd &= 2\pi R2
 \end{aligned} \tag{6.1}$$

Wd will be of size the integer part of the resulting size (pixels are concrete values no decimals). Using all these parameters it is possible map each image point into a rectangular view using the following equations (Equation 6.2):

$$\begin{aligned}
 r &= \left(\frac{y}{Hd}\right)(R2 - R1) + R1 \\
 \theta &= \left(\frac{x}{Wd}\right)2\pi \\
 xs &= Cx + r \sin \theta \\
 ys &= Cy + r \cos \theta
 \end{aligned} \tag{6.2}$$

Figure 6.13 shows some examples of different fisheye images with persons in different parts of the monitored room. At the bottom, the unrolled images are presented where some interesting cases happen. In the first example persons are now in the same perspective. In the second image it is possible see that persons far from the center of the camera have been mapped in a frontal view but persons exactly in the middle of the cut radius appear in both left and right borders of the image. Third and fourth examples show problems that appear in these cases when persons are under the camera or very near to the deleted part of the image (internal selected circle).

In the other hand, 360° images will be treated in a different way. In first instance, the data collection will be using the Equirectangular image Projection that is suitable to represent the entire environment in only one rectangular distorted image (Figure 6.14 bottom image). The problem of this image is the inherent distortion in the top and bottom parts. Depending on the application this can affect dramatically the detection process needing a different way to represent the 360° information. In this work, Cubemap projection have been used along Equirectangular projection to compare the performance of the detector.

Figure 6.13: Examples of Fisheye and unrolled images.

The cubemap projection can be seen as the composition of 6 N FoV projections from the equirectangular image where the horizontal and vertical field of view is 90° covering the entire image. The N FoV images can be obtained using the Gnomonic Projection that is a map projection which can be achieved by projecting the ray of light from center of the sphere through the surface of the sphere at point P1, which touches on a plane at another point P where the plane is tangent to the sphere at point S as shown in Figure 6.15). This mapping of point from P1 to P on the surface is called Gnomonic map projection.

Figure 6.14: *Left:* Equirectangular projection of a 360° images. *Right:* Cubemap projection obtained from the Equirectangular image.

Figure 6.15: Intuition of the sphere to plane projection and example of NFoV image extracted from the equirectangular image.

At the tangent point there is zero distortion, but it increases as we go away from it. It projects great circles on straight lines. It can project a hemisphere of the sphere on the plane. The Gnomonic projections are used for extracting a NFoV from an equirectangular image. Since an equirectangular image is an unwrapped version of the sphere, the definition and mathematics of Gnomonic projection states the same.

Let us consider the point S in Figure 6.15 to have the central latitude and central normalized longitude of $\lambda_0, \phi_1 = (0, 0)$ which is on the equator and also the center of the plane. The transformation equations of the plane coordinates are given by the following Equations 6.3 and 6.4:

$$\begin{aligned} x_{NFoV} &= \frac{\cos \phi \sin (\lambda - \lambda_0)}{\cos c} \\ y_{NFoV} &= \frac{\cos \phi_1 \sin \phi - \sin \phi_1 \cos \phi \cos (\lambda - \lambda_0)}{\cos c} \end{aligned} \quad (6.3)$$

where c is angular distance of point x, y from the center of the projection given as:

$$\cos(c) = \sin \phi_1 \sin \phi + \cos \phi_1 \cos \phi \cos (\lambda - \lambda_0) \quad (6.4)$$

Using the above equations, a point can be mapped in the plane selecting the horizontal and vertical view angle we want extract and the center projection point. To perform the opposite operations a point can be translated from the plane to the sphere using Equations in 6.5:

$$\begin{aligned} \phi_{sph} &= \sin^{-1}(\cos(c) \sin \phi_1 + \frac{y_{NFoV} \sin(c) \cos \phi_1}{\rho}) \\ \lambda_{sph} &= \lambda_0 + \tan^{-1}(\frac{x_{NFoV} \sin(c)}{\rho \cos \phi_1 \cos(c) - y_{NFoV} \sin \phi_1 \sin(c)}) \\ \rho &= \sqrt{x_{NFoV}^2 + y_{NFoV}^2} \\ c &= \tan^{-1} \rho \end{aligned} \quad (6.5)$$

Finally, equirectangular projection can be also converted in a dual fisheye image. In this way it is possible represent the information as in a Fisheye image but having two of these

cameras at the same time pointing to opposite views. The equations to convert between equirectangular projection to side by side fisheye projection and viceversa are presented step by step. Firstly, 2D normalized pixel points from the equirectangular projection are converted in 3D coordinate space using latitude and longitude:

$$\begin{aligned} longitude &= \pi x_{rect} \\ latitude &= \frac{\pi y_{rect}}{2} \end{aligned} \quad (6.6)$$

Each point can be represented in the 3D space taken these equations as follows:

$$\begin{aligned} P_x &= \cos(latitude) \cos(longitude) \\ P_y &= \cos(latitude) \sin(longitude) \\ P_z &= \sin(latitude) \end{aligned} \quad (6.7)$$

Finally, based on this new coordinate system each point can be reprojected to the fisheye projection using the following equations:

$$\begin{aligned} r &= \frac{2 \operatorname{atan2}(\sqrt{P_x^2 + P_z^2}, P_y)}{aperture} \\ \theta &= \operatorname{atan2}\left(\frac{P_z}{P_x}\right) \\ x_{fish} &= r \cos \theta \\ y_{fish} &= r \sin \theta \end{aligned} \quad (6.8)$$

Where *aperture* means the camera FoV in the case of don't want recover equirectangular image is 180°. If a reprojection from fisheye to equirectangular is needed aperture should be a value of at least 10 degrees more than 180° is required to have sufficient blend zone to combine both fisheye projections in the final equirectangular panorama image using, for example, a simple linear multiplicative blend function. The equations to recover the equirectangular

projection are the opposite process. Firstly, 2D fisheye pixel points are converted in a 3D vector:

$$\begin{aligned}\phi &= \frac{r \cdot aperture}{2} \\ \theta &= atan2(y_{fish}, x_{fish})\end{aligned}\tag{6.9}$$

Secondly, the 3D vector is converted to latitude and longitude by the following equations:

$$\begin{aligned}longitude &= atan2\left(\frac{P_y}{P_x}\right) \\ latitude &= atan2(P_z, \sqrt{P_x^2 + P_y^2})\end{aligned}\tag{6.10}$$

Finally, the normalized pixel coordinate points are obtained converting latitude and longitude to the final coordinate:

$$\begin{aligned}x_{rect} &= \frac{longitude}{\pi} \\ y_{rect} &= 2\frac{latitude}{\pi}\end{aligned}\tag{6.11}$$

An example of this projection is presented in Figure 6.17 where an equirectangular projection have been converted in a dual fisheye with an aperture of 180° is presented.

Figure 6.16: Visual explanation of Equirectangular/Fisheye to Fisheye/Equirectangular projection respectively.

Figure 6.17: Example of equirectangular image converted to dual fisheye with 180° of aperture in both sides.

To finish this part, depending on the camera position a problem can appear in the images. For example, regarding Figure 6.18 it is shown that persons can appear in the center of the image or totally rotated in the sphere if camera is not perpendicular to the floor. Rotations in the sphere can be applied to put the image in a way where the person can be detected accurately without or with less distortions. This transformation can be carried out with these steps:

- Translate the x and y position of each pixel (in the equirectangular projection) into a longitude, $-\pi \leq longitude \leq \pi$, and a latitude, $-\pi/2 \leq latitude \leq \pi/2$. This is a linear transformation.
- Create x , y , and z coordinates of the corresponding longitude and latitude on a unit sphere where positive z is the north pole (normalized coordinates).
- Rotate these Cartesian coordinates in the sphere coordinate system.
- Calculate the longitude and latitude of the rotated coordinates.
- Transform the new longitude and latitude to a pixel position.

Figure 6.18: Examples of same equirectangular image with the camera rotated in different axis.

All these steps can be carried out using previous equations to convert between equirectangular and latitude, longitude. The rotation is a general transformation that is performed using the sphere transformation matrix in spherical coordinates. Finally, the new rotated longitudes and latitudes can be projected again in the equirectangular image.

6.5 Camera calibration

with the introduction of the cheap pinhole cameras they became a common occurrence in our several work proposals. Unfortunately, this cheapness comes with its price producing a significant distortion or having cameras with lenses that directly map the image with a concrete distortion. Luckily, these are constants and with a calibration and some remapping it is possible correct this problem. Additionally, with calibration can be also determined the relation between the camera's natural units in pixels and the real world units in millimeters

for example.

Geometric camera calibration is a method to estimate the parameters of a lens and the image sensor of an image or video camera. Using these parameters lens distortions can be corrected, the size of an object in world units can be computed and also the location of the camera in the scene can be calculated. The main goal in this work is convert the 2D pixel coordinate system to a real 3D scenario to match the image detections in the 3D world.

For this purpose, the known pinhole camera model is used and an illustration of this model is drawn in Figure 6.19. The way an image is created by a system of lenses is similar to the process followed by the human eye. It is equivalent to making the projection in linear perspective on an appointed plane (the plane of the camera sensor). The model consists of an opaque plane, with a very small hole at its center and the image plane is parallel to the mentioned plane at f distance. A point in the surface of an object, in the tree-dimensional world has $P(X, Y, Z)$ coordinates and the projection of P on the image plane will be another point $p(x, y)$ with Z continuous coordinate, and with coordinate $Z = 0$ in the reference system.

Figure 6.19: Pinhole camera model.

This model can be represented by the following Equation 6.12:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (6.12)$$

Where X, Y, Z are the coordinates of a 3D point in the world coordinate space, x, y are the coordinates of the projection point in pixels, first matrix is a camera matrix, or a matrix of intrinsic parameters with cx, cy the principal point that is usually at the image center and fx, fy are the focal lengths expressed in pixel units. The r_{aa} and t_a represent the rotation and translation parameters also known as the extrinsic camera matrix.

For the distortion, normally, the radial and tangential factors are taken into account. For the radial factor the following model is used (Equation 6.14):

$$x_{distorted} = x(1 + k_1r^2 + k_2r^4 + k_3r^6) \quad (6.13)$$

$$y_{distorted} = y(1 + k_1r^2 + k_2r^4 + k_3r^6) \quad (6.14)$$

Then for an undistorted pixel point at (x, y) coordinates, its position on the distorted image will be $x_{distorted}, y_{distorted}$. The presence of the radial distortion manifests in form of the *Barrel* or *Fisheye* effect.

Tangential distortion occurs because the image taking lenses are not perfectly parallel to the imaging plane. It can be represented via the formulas (Equation 6.16):

$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)] \quad (6.15)$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2xy] \quad (6.16)$$

So in total there are five distortion parameters involved in the system (Equation 6.17):

$$distortion_{coefficients} = (k_1, k_2, p_1, p_2, k_3) \quad (6.17)$$

The distortion coefficients do not depend on the scene viewed. Thus, they also belong to the intrinsic camera parameters and they remain the same regardless of the captured image resolution scaling appropriately. An example of the most common distortions is shown in Figure 6.20:

Figure 6.20: *Left:* Image with no distortions. *Center:* Image with Radial distortion. *Right:* Image with Tangential distortion

The parameter w is explained by the use of homography coordinate system (and $w=Z$). The steps involved to obtain the intrinsic and extrinsic camera parameters and the distortion coefficients is presented in [266] and combined with [267]. The coordinates of 3D object points and their corresponding 2D projections in each view must be specified. That may be achieved by using an object with a known geometry and easily detectable feature points. Such an object is called a calibration rig or calibration pattern, and a chessboard is a common calibration rig to use. The algorithm performs the following steps:

1. Compute the initial intrinsic parameters. The distortion coefficients are all set to zeros initially.
2. Estimate the initial camera pose as if the intrinsic parameters have been already known.
3. Run the global Levenberg-Marquardt optimization algorithm to minimize the reprojection error, that is, the total sum of squared distances between the observed feature points and the projected (using the current estimates for camera parameters and the poses) object points.

The algorithm optimizes the distances reducing the reprojection error to a concrete level after several iterations using the provided calibration points in the rig pattern. After the optimization process all the necessary camera parameters are estimated, intrinsic and extrinsic camera matrices and distortion coefficients. Now, using this information a new set of points must be collected in the distorted or undistorted image and the second step of the previous algorithm is used to estimate the object pose given the collected points at unknown locations

and the previous estimated parameters (intrinsic camera matrix and distortion coefficients) to find the rotation and translation vectors.

The rotation vector is firstly transformed into a rotation matrix and then concatenated with the translation vector to get the extrinsic camera matrix and finally using Equation 6.12 the conversion between 2D and 3D is achieved.

An example of calibration images taken from the NFoV and Fisheye camera is shown in Figures 6.21 and 6.22 respectively. In these images a chessboard pattern was used. 18 images were captured with the pattern in different positions to cover the major number of possibilities around the entire image. The NFov camera due to its large horizontal FOV important distortions at the left and right parts appear but not severe. In contrast, for the fisheye camera it is shown an important degree of distortion due to the lens used.

Figure 6.21: Example of NFoV images taken with the chessboard pattern in different positions of the image for calibration.

Figure 6.22: Example of Fisheye images taken with the chessboard pattern in different positions of the image for calibration.

Finally, Figures 6.23 and 6.24 shown the achieved undistorted images using the collected calibration images with the chessboard pattern. In the first case a positive radial distortion is presented in the image and after undistort the image in Figure 6.23 some black parts appear due to the lens distortion correction. Now, in these images the chessboard pattern is totally rectified. In the case of the fisheye camera the same happens but in a different way (Figure 6.24). The chessboard pattern has been rectified and now the entire room look as a rectangle without the initial circular distortion. The main problem after the undistortion in this image is that higher parts of the room look expanded and depending on the application this situation can present a problem. In this work, this process is used to calibrate the camera and can convert effectively from the 2D pixel world to the 3D world.

Figure 6.23: NFOV Undistorted

Figure 6.24: Zenith Undistort

To fix a concrete camera coordinates system instead of use the estimated one it is necessary take some points at real locations in the image and use the pixel coordinates and the 3D world real coordinates. RANSAC [268] algorithm is used to estimate the object pose given a set of object points, their corresponding image projections, as well as the camera matrix and the distortion coefficients. This algorithm finds such a pose that minimizes reprojection error, that is, the sum of squared distances between the observed projections (image points) and

the projected object points. The use of RANSAC makes the function resistant to outliers.

After these algorithms a new extrinsic parameters in the new coordinate system are obtained. The final X, Y coordinates in the world coordinate system can be extracted directly from Equation 6.12 solving to get the last vector and using a input the tracking point used to track the person.

For the NFoV camera the undistorted image is used to get the estimated tracking point which will be the center of the bounding box. In the case of Fisheye camera, the tracking points in the image will be the center of the bounding box and will pass firstly by the undistort and the undistorted point will be used to get the world coordinates. Finally, in the 360° camera the double fisheye version is used firstly to calibrate both lens of the camera and secondly to undistort one of the fisheye images were the tracking point (center of the bounding box) is. An example of the Fisheye process to get the world coordinates is showed in Figure 6.25.

Figure 6.25: Process to convert between pixel coordinates to real world coordinates. *Left:* Original image, *Center:* Undistorted image, *Right:* Selected points in the image for calibration.

6.6 Image person detection

One of the main important tasks in image processing and computer vision is the detection and recognition part. Normally, the starting point of a computer vision system need detect the location of some concrete objects or recognize at a concrete moment what is the current image. In this work the most extended state-of-the-art object detectors models have been used an retrained in a new dataset for person detection refinement.

6.6.1 Object detection algorithms

YOLO is the first object detector tested. This algorithm has been improved from the version 1 to version 3 (the latest). Main modification have been in the architecture (Figure 6.26) introducing batch normalization, higher resolution classifier (from 224×224 to 416×416 image size), introduction of anchor boxes, fine-grained features for small objects, multi-scale training, Darknet53 backbone, and a Feature Pyramid Network (FPN) to compete with the best solution at the moment which is Faster-RCNN.

11

Figure 6.26: YOLOv3 network architecture.

Prior detection systems re-purpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections. In this a single neural network is applied to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

The model has several advantages over classifier-based systems. It looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast 7x compared with Faster-RCNN.

The second proposal is the use of Faster-RCNN or in its last version, Mask-RCNN which contains a new branch for image segmentation but the previous branches work in the same way as in the previous version 6.27. In this architecture selective search algorithm (slow and time-consuming process) is eliminated letting the network learn the region proposals. Similar to Fast R-CNN [166], the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. In comparison with YOLOv3, this network is slow (12-15 FPS) but is extremely accurate thanks to the Region Proposal Network (RPN) module.

Figure 6.27: Mask-RCNN network architecture.

Next algorithm is the Single Shot Detector (SSD). Key idea is to use a single network 6.28 (for speed) and no need for region proposals, instead it uses different bounding boxes and then adjust the bounding box as part of prediction. Different bounding box predictions is achieved by each of the last few layers of the network responsible for predictions for progressively smaller bounding box and final prediction is union of all these predictions. SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage.

The main improvement is in the inclusion of a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales.

Figure 6.28: Single Shot Detector network architecture.

It is discovered that there is extreme foreground-background class imbalance problem in one-stage detector. And it is believed that this is the central cause which makes the performance of one-stage detectors inferior to two-stage detectors. In RetinaNet network 6.29, a one-stage detector, by using focal loss, lower loss is contributed by “easy” negative samples so that the loss is focusing on “hard” samples, which improves the prediction accuracy. With ResNet+FPN as backbone for feature extraction, plus two task-specific subnetworks for classification and bounding box regression, forming the RetinaNet, which achieves state-of-the-art performance,

outperforms Faster R-CNN, the well-known two-stage detectors.

Figure 6.29: RetinaNet network architecture.

Finally, following how Fast R-CNN improved on the originals detection speed by sharing a single CNN computation across all region proposals, R-FCN increase speed by maximizing shared computation with a similar idea. In this way R-FCN, or Region-based Fully Convolutional Network (Figure 6.30), shares 100% of the computations across every single output, being fully convolutional running into a unique problem in model design.

Position-sensitive score maps are required and represent one relative position of one object class. A score map might activate where it sees some parts of the object. Essentially, these score maps are convolutional feature maps that have been trained to recognize certain parts of each object. A CNN is applied as backbone over the input image and a fully convolutional layer generate a score bank of the aforementioned “position-sensitive score maps”.

R-FCN considers each region proposal, divides it up into sub-regions, and iterates over the sub-regions. It repeats this for all possible classes. If enough of the sub-regions are parts of a same object, the RoI gets classified as the object after a softmax over all the classes. With this setup, R-FCN is able to simultaneously address location variance by proposing different object regions, and location invariance by having each region proposal refer back to the same bank of score maps. As a result, R-FCN is several times faster than Faster R-CNN, and achieves comparable accuracy.

Figure 6.30: Region Fully Convolutional Network architecture.

All these algorithms have been tested in order to check the performance of the object detectors in terms of accuracy and speed due to real time processing is required for the proposed final architecture. All the networks were pretrained on COCO dataset 2017 version. In order to improve the performance in person detection (due to the algorithms are trained for several classes) all the networks have been retrained using PASCAL-VOC dataset using only person instances. As validation data, 100 video recordings in the proposed scenario of 30 seconds have been recorded and labeled (with bounding boxes for detection and tracking checking and person IDs to validate reidentification algorithms) where between 3 and 8 persons are walking in a normal situation along the entire monitored scenario.

6.7 Person reidentification

Person reidentification is one topic that attracts research community due to the high complexity involved in this task. It consists on identify the same person along time in one or multiple cameras. There are some interesting proposals that try to exploit more complex feature using concrete person characteristics like clothes and body parts. Common pretrained convolutional neural networks (VGGx, ResNetx, SqueezeNet...) can't focus the attention in more concrete parts of the image extracting a more general feature vector that sometimes is not so discriminative to classify same persons across an entire test dataset.

To deal with this problem a big amount of datasets is available with different characteristics. Normally the datasets contains directly the cut persons from the image in several different poses (front, back to the camera for example) and some of them with high light changes.

Other datasets contains the cutted tracking of the person in video recordings. Finally, video datasets with tracked person annotated are also available.

Market 1501 dataset [269] contains over 32,000 annotated bounding boxes, plus a distractor set of over 500K images and was collected in an open system where each identity has multiple images under each camera. It contains a large number of identities and each identity has several images from six dis-joint cameras. This dataset also includes 2793 false alarms from DPM as distractors to mimic the real scenario. Quality of the bounding boxes is worse than CUHK03. Later in the ICCV 2015 release version, 500K distractors are integrated to make this dataset really large scale. In the original paper proposed this dataset, the author also used mAP as an evaluation criteria to test the algorithms.

CUHK03 dataset [213] with 13,164 images of 1,360 pedestrians. CUHK03 is the first person re-identification dataset that is large enough for deep learning. It provides the bounding boxes detected from deformable part models (DPM) and manually labeling. Person detection quality is relatively good for this dataset.

DukeMTMC [270] using 8 disjoint surveillance camera views covering parts of the Duke University campus. The DukeMTMC dataset is a large-scale heavily labeled multi-target multi-camera tracking dataset. In total, more than 2700 people were labeled with unique identities in 8 cameras. With the access to all information (full frames, frame level ground truth, calibration information, etc.), this dataset has a lot of potentials. Based on the released train-validation set, two re-id extension datasets are created. The key difference is the way to generate the bounding boxes. DukeMTMC-reID directly uses the manually labeled ground truth whereas DukeMTMC4ReID adopts Doppia as the person detector.

MSMT17 [271] This large scale re-id dataset is collected in a campus with 12 outdoor cameras and 3 indoor cameras. It covers 4 days with different weather in a month. For each day, 3 one-hour videos are selected from morning, noon and afternoon. Faster RCNN is utilized for pedestrian detection. This dataset is the largest re-id dataset so far. It has similar viewpoint with Market, but much more complicated scenarios.

VIPeR [272] This dataset contains two cameras, each of which captures one image per person. It also provides the viewpoint angle of each image. Although it has been tested by many researchers, it's still one of the most challenging datasets.

PRID dataset [273] has 385 trajectories from camera A and 749 trajectories from camera B. Among them, only 200 people appear in both cameras. This dataset also has a single

shot version, which consists with random selected snapshots. Some trajectories are not well-synchronized, which means the person might "jump" between consecutive frames.

The MARS (Motion Analysis and Re-identification Set) dataset [274] is an extension version of the Market1501 dataset. It is the first large scale video-based person re-id dataset. Since all bounding boxes and tracklets are generated automatically, it contains distractors and each identity may have more than one tracklets. Precomputed deep features are also available on the website.

[250] Based on the assumption that the real person re-identification system should have the trajectory for each identity, iLIDS-VID dataset extracted 600 trajectories for 300 identities from iLIDS MCTS dataset. Due to the limitation of iLIDS MCTS dataset, iLIDS-VID has extremely heavy occlusion.

More characteristics and example of images in some of these datasets are presented in Table 6.1 and Figure 6.31 respectively.

Table 6.1: Comparison between most used reidentification datasets

Dataset	# identities	# cameras	# images	Label method	Crop size	Multi shot	Track sequences	Full frames
Market 1501	1501	6	32217	Hand/DPM	128X64	✓		
CUHK03	1467	10(5 pairs)	13164	Hand/DPM	Vary	✓		
DukeMTMC	1812	8	36441	Hand	Vary	✓		✓
MSMT17	4101	15	126441	Faster-RCNN	Vary	✓		
ViPeR	632	2	1264	Hand	128X48			
PRID2011	934	2	24541	Hand	128X64	✓	✓	✓
MARS	1261	6	1191003	DPM+GMMCP	256X128	✓	✓	
iLIDS-VID	300	2	42495	Hand	Vary	✓	✓	

In this work four reidentification algorithms have been tested and trained in several datasets at the same time and using a custom dataset with images collected in the real monitored scenario. These algorithms are MuDeep [256], HACNN [224], PCB [220] and MLFN [235].

The differences in the person appearance are often subtle and only detectable at the right location and scales. Normally, older re-id models, particularly the recently proposed deep learning based (using known networks like VGGx, ResNetx between among others) one's match people at a single scale. In order to focus the attention in different image parts multi-

scale deep learning model is proposed in[256] named MuDeep.

Figure 6.31: Example of images in different reidentification datasets.

This model is able to learn deep discriminative feature representations at different scales and automatically determine the most suitable scales for matching. Furthermore, the importance of different spatial locations for extracting discriminative features is also learned explicitly. Figure 6.32 shows an example of how human can identify concrete human parts to identify

the person and how the network should work to achieve a similar result. The architecture has several steps composed by different branches with more or less depth to focus the importance in big and little body parts.

Figure 6.32: *Left:* MuDeep algorithm intuition. *Right:* MuDeep multiscale network architecture

In person re-identification methods is assumed the availability of well-aligned person bounding box images as model input or rely on constrained attention selection mechanisms to calibrate misaligned images. This is an initial wrong assumption regarding real world application decreasing the performance of the reidentification matching due to the big amount of pose variations and detection errors.

The model proposed in [224] is called HACNN and learn at the same time inside the network an attention selection and feature representation in a Convolutional Neural Network (CNN) by maximizing the complementary information of different levels of visual attention. The model learns soft pixel attention and hard regional attention along with simultaneous optimization of feature representations, dedicated to optimize person re-id in uncontrolled (misaligned) images. Figure 6.33 represents the network architecture of HACNN model where attention modules are shared between different network branches to focus the importance of the learning in different discriminative image parts.

Figure 6.33: HACNN network architecture where attention modules are computed after Inception networks blocks and shared between different image representations along the network.

Employing part-level features offers fine-grained information for pedestrian image description offering a powerful tool to focus the learning in concrete person body parts. A prerequisite of part discovery is that each part should be well located.

In PCB model [220] instead of use external resources like pose estimator, we authors consider a content consistency within each part for precise part location having as previous knowledge that some body parts should be in the same part of the image in a normal person walking situation. They propose learn discriminative part-informed features for person retrieval proposing a network which given an input image, it outputs a convolutional descriptor consisting of several part-level features with a uniform partition strategy. Finally, a refined part pooling method re-assigns outliers to the parts they are closest to, resulting in refined parts with enhanced within-part consistency. Figure 6.34 shows the network structure where the uniform partitioning is represented.

Figure 6.34: PCB network architecture where the uniform partitioning is showed.

Finally, modelling discriminative and view-invariant factors of person appearance at both high and low semantic levels is the key for effective person reidentification. Deep learning extended model focus the attention in the entire image and other recent works try to propose laborious human annotated parts in the image to be used in the learning process.

MLFN network [235] factorizes the visual appearance of a person into latent discriminative factors at multiple semantic levels without manual annotations. The network is composed of multiple stacked blocks that contains multiple factor modules to model latent factors at a specific level, and factor selection modules that dynamically select the factor modules to interpret the content of each input image. Figure 6.35 shows the multiple blocks to extract separated rich features between network blocks and combine the information to be the input of the next layers.

Figure 6.35: MLFN network architecture where multiple blocks extract independent features that are combined to be the input of more deeper layers.

6.8 Image person tracking

In order to connect the detections between frames a tracking is needed over the detected objects. These types of algorithms are able to link detections between consecutive frames. In this case and in comparison with the tracking algorithms presented in Chapter 5 working with wireless and inertial sensors it is possible not only take into account the center of the bounding boxes as the main point to track. In image-based detection and tracking algorithms like Kalman or Particle filters can be used as in the previous proposed wireless and inertial tracking system but including more equations or expanding the number of parameters including not only the center point of the bounding box also the width and height can be tracked and estimated.

The approach presented in the last paragraph is a first attempt to achieve a good tracking performance depending on several aspects. The first point is related to the fact that occlusions can happen during video recordings. This problem can affect severely the performance of the tracker incurring in errors in the estimations. Secondly, another problem is related to the tracker time to live after loss detections. When a person is being tracked by the algorithm and for example an occlusion appear losing the tracker the algorithm try to estimate the following states without information about new measurements. After the occlusion the tracker can appear again and depending on the tracking accuracy and the total time where the tracker was lost it is possible recover the tracking or loss complete the tracker and assign a new tracker id to the same person. Finally, the last problem is related to multitasking in one image. Tracked objects that are very near can confuse the tracker id and change or lost the own id with another one changing completely the real trajectory of the initial person.

To reduce the effect of these problem it is possible not only take the bounding box information as the only source to compute the next tracking states. Working with images some features about the image and the objects can be extracted to match better consecutive frames in time. This is how some image-based tracking algorithms works. In the case of no deep learning algorithms to perform object tracking, in this work, KCF, CSRT and Mosse algorithms have been tested under the conditions of the proposed scenario. To have a good comparison between no learning and learning algorithms Goturn, Re3, Siamese-CNN and Deep-Sort algorithms have been employed in the same conditions. The last algorithms tested was Detect & Track which is an algorithm that joint the detections and tracking in a common framework performing both operations with only one deep learning network.

6.8.1 Visual online tracking using image information

An object to be tracked is usually selected by a rectangular bounding box. The task of a tracker is to follow the object in the video by updating the bounding box parameters (the position at the simplest case). Correlation filter-based tracking comprises the following steps:

- Assuming a template of a target object T and an input image I , the Fast Fourier Transform (FFT) of both the template (T) and the image (1) is computed.
- A convolution operation is performed between template T and image 1.
- The result from previous step is inverted to the spatial domain using Inverse Fast Fourier Transform (IFFT). The position of the template object in the image I is the max value of the IFFT response we get.

This correlation filter-based technique has limitations in the choice of T . As a single template image match may not observe all the variations of an object, such as rotation in the image sequence then the following tracking proposals take the advantage of correlation filters with different modifications.

The basic idea of the correlation filter tracking [275] is estimating an optimal image filter such that the filtration with the input image produces a desired response. The desired response is typically of a Gaussian shape centered at the target location, so the score decreases with the distance. The filter is trained from translated (shifted) instances of the target patch. When testing, the response of the filter is evaluated and the maximum gives the new position of the target. The filter is trained on-line and updated successively with every frame in order the tracker adapts to moderate target changes.

Major advantage of the correlation filter tracker is the computation efficiency. The reason is that the computation can be performed efficiently in the Fourier domain. Thus the tracker runs super-realtime, several hundred frames per second (FPS). There are both linear and non-linear (kernel) versions of the tracker that are derived from a unifying least square principle.

The main advantages of this tracker are that is very fast, can adapts it to scale and rotation variances is online trained on a single image patch, provides an aggressive failure reporting, all the parameters are manually adjustable by the user and supports custom feature extractors. In contrast the most important disadvantages are a not recover from failures well (it is preferable

start again the algorithm after a failure report), does not recover when objects are changed out of view, algorithm cannot recover the state from multiple consecutive failures and finally not incorporate motion into estimation neglecting some motion patterns that cannot be estimated using only the features information.

In the Discriminative Correlation Filter with Channel and Spatial Reliability (CSRT) [144], it is used the spatial reliability map for adjusting the filter support to the part of the selected region from the frame for tracking. This ensures enlarging and localization of the selected region and improved tracking of the non-rectangular regions or objects. It uses only two standard features (HoGs and Colornames). Reliability scores reflect channel-wise quality of the learned filters and are used as feature weighting coefficients in localization. It also operates at a comparatively lower fps (25 fps) but gives higher accuracy for object tracking.

The pros using this tracker are that is slower but more accurate than KCF, provide a robust to unpredictable motion is trained on a single image patch (equally to KCF), the algorithms can recover the state from failures when the object hasn't moved much, can tolerate intermittent frame drops, it reports unrecoverable failures, is able to adapt it to scale, deformation and rotation and the parameters are selected adjustable manually. In the other hand the cons of the algorithms are not well recovering from failures due to full occlusion, latches appear onto surrounding regions when partially occluded resulting in drift, does not recover when objects are changed out of view, does not recover from multiple consecutive failures, does not incorporate motion into estimation

Minimum Output Sum of Squared Error (MOSSE) [147] uses adaptive correlation for object tracking which produces stable correlation filters when initialized using a single frame. MOSSE tracker is robust to variations in lighting, scale, pose, and non-rigid deformations. It also detects occlusion based upon the peak-to-sidelobe ratio, which enables the tracker to pause and resume where it left off when the object reappears. MOSSE tracker also operates at a higher fps (450 fps and even more). To add to the positives, it is also very easy to implement, is as accurate as other complex trackers and much faster. But, on a performance scale, it lags behind the online and deep learning based trackers in accuracy.

Finally a different approach explores a pragmatic idea to multiple object tracking where the main focus is to associate objects efficiently for online and realtime applications. The algorithm is known as Simple Online Real-Time Tracker (SORT) [276]. To this end, it is known that detection quality is identified as a key factor influencing tracking performance. It is possible use only some rudimentary combination of very simple and known techniques such as the Kalman Filter (for tracking) and Hungarian algorithm (for data association) for

the tracking components achieving an accuracy comparable to state-of-the-art online trackers. Furthermore, due to the simplicity of this tracker tracking method, the tracker updates at a very fast rate of more than 200 FPS.

6.8.2 Visual tracking with Deep Learning algorithms

The problem of arbitrary object tracking has traditionally been tackled by learning a model of the object's appearance exclusively online, using as sole training data the video itself. In contrast, Deep Learning approaches are gaining an important attention in this field. Many real-time trackers rely on online learning algorithms that are typically much faster than a Deep Learning based solution using more simple operations. The first algorithms with interesting results using deep learning approaches was GOTURN [185], short for Generic Object Tracking Using Regression Networks. Most tracking algorithms are trained in an online manner. In other words, the tracking algorithm learns the appearance of the object it is tracking at runtime. GOTURN changed the way that Deep Learning is applied to the problem of tracking by learning the motion of an object in an offline manner. The GOTURN model is trained on thousands of video sequences and does not need to perform any learning at runtime. The algorithm is trained using a pair of cropped frames from thousands of videos.

Figure 6.36: GOTURN tracker architecture.

In the first frame (referred as the previous frame), the location of the object is known, and the frame is cropped to two times the size of the bounding box around the object. The object in the first cropped frame is always centered. The location of the object in the second frame (current frame) needs to be predicted. The bounding box used to crop the first frame is also used to crop the second frame. Because the object might have moved, the object is not centered in the second frame. A Convolutional Neural Network (CNN) is trained to predict the location of the bounding box in the second frame.

Both frames pass through a bank of convolutional layers. The layers are simply the first five convolutional layers of the CaffeNet architecture. The outputs of these convolutional layers are concatenated into a single vector of length 4096. This vector is input to 3 fully connected layers. The last fully connected layer is finally connected to the output layer containing 4 nodes representing the top and bottom points of the bounding box.

In addition, another interesting work take the advantage of fully convolutional neural networks using the concept of Siamese Networks [277]. The output of this approach is a scalar-valued score map whose dimension depends on the size of the search image. This enables the similarity function to be computed for all translated sub-windows within the search image in one evaluation.

Figure 6.37: Architecture of the Fully-Convolutional Siamese Network for object tracking.

The main idea is learn using a similarity function how a part of the image is close similar to the previous searching region to track. In comparison with GOTURN this model is more robust to fast changes and the network learns to minimize the error between frames. To train the network the input is composed by a test image and the complete image. The algorithms learn to find the most similar region in the entire image to the proposed cut region from a previous frame. The model achieves between 50 and 80 fps using GPU, enough velocity to

work in real time having elevate accuracy detections.

Finally, Re3 [184] is a real-time deep object tracker capable of incorporate temporal information into the model. Instead of focusing on a limited set of objects or training a model at test time to track a specific instance, the algorithm is pretrained on a large variety of objects and is efficiently updates on the fly simultaneously tracking and updating the appearance model with a single forward pass. This model is capable of tracking objects at 150 FPS.

challenging benchmarks. We also show that our method handles temporary occlusion better than other comparable trackers using experiments that directly measure performance on sequences with occlusion.

Figure 6.38: Architecture of the Re3 network for object tracking.

The Network Structure is shown in Figure 6.38 where image crop pairs are fed in at each timestep. Both crops are centered around the object's location in the previous frame, and padded to two times the width and height of the object. Before every pooling stage, a skip layer is added to preserve high-resolution spatial information. The weights from the two image streams are shared and the output from the convolutional layers feeds into a single fully connected layer and an LSTM (to learn in time) predicting the bounding box coordinates.

It can be observed that the three proposals use the information from the previous and the current frame in order to learn a motion pattern between frames to predict effectively the next state of the tracker. Problems related to this proposal are that all the algorithms are for single object tracking and for achieve multiple object tracking you have to run each algorithm over each object that you want track losing speed in the prediction. Another problem is that the first bounding box have to be given to the network as initial state of the object in the first frame needing the addition of an initial stage of object detection that can be performed using some of the proposed algorithms in previous sections.

6.8.3 Mixing detection, tracking and reidentification

The previous sections show a set of algorithms to perform detection, tracking and reidentification separately. There exists now and big effort to combine the algorithms in order to have a unique framework that perform at least two of the three steps in a single algorithm step.

One of these new proposals is Deep-SORT [278] which combines object detection using YOLOv3 in combination with the SORT algorithm to perform detection and tracking in a single algorithm step (Figure 6.39). This algorithm is not a purely unique framework where the network can learn to perform both operations at the same time. In the algorithm pre-trained YOLOv3 is used to extract candidate bounding boxes that are the input to the SORT algorithm. After the association, a final step is performed using a Deep Neural Network trained in reidentification datasets in order to extract rich features to associate actual trackers and lost trackers in the same image with no only tracking information, but also using image features. The association part combines the powerful of CNNs for reidentification feature extraction and the tracking association in a single step. The problems with this approach is the processing time reducing the FPS to around 12-20 using GPU.

Figure 6.39: Deep-SORT architecture composed by an initial person detector and a multi tracking algorithm using Kalman filter with a deep association metric.

A different approach is a purely Deep Learning solution to combine detections and tracking in a single framework which is able to learn to perform both ideas at the same time. This approach is called Detect and Track [192] and the network architecture is presented in Figure 6.40, where two branches take as input the previous and the current frame to finally output the bounding boxes for each detected object (Figure 6.40).

Figure 6.40: Detect and Track network architecture for object detection and tracking in a single framework.

A Convolutional Network architecture use a multi-task objective for frame-based object detection and across-frame track regression is able to learn both detection and tracking at the same time. Correlation features (something similar to Correlation Features presented in on-line trackers) that represent object co-occurrences across time are used to include information in time and finally the algorithm link the frame level detections based on the across-frame tracklets to produce high accuracy detections at the video level. Due to the complexity of the network and the correlation features the network can provide a FPS of around 7-9 losing completely the real time capabilities.

Finally, the last tested algorithm is called AlphaPose and is a complete framework for body keypoint detection and person detection and tracking. The first version [180] was able to detect persons and estimate the body keypoints 5 times faster than its main competitor OpenPose [279]. Some months later a new version included a new framework named PoseFlow [280] which is able to track poses efficiently at 10 FPS.

Figure 6.41: Alpha Pose architecture with spatial transformer and SPPE network for body keypoint detection.

Human detectors have demonstrated good accuracy and performance but small errors in localization and recognition are inevitable in the wild. These errors can cause failures for a single person pose estimator, especially for methods that solely depend on human detection results. In this algorithm it is proposed a novel regional multi-person pose estimation framework to facilitate pose estimation in the presence of inaccurate human bounding boxes (Figure 6.41). Additionally the authors propose a decent and efficient pose tracker based on pose flows composed by an online optimization framework to build the association of cross-frame poses and form pose flows and a pose flow non-maximum suppression which is capable to robustly reduce redundant pose flows and re-link temporal disjoint ones showing high accuracy in challenging tracking datasets as MPII [281].

All these frameworks have something very important in common, the processing time is high. Due to this main limitation no real time solution is possible in a real scenario like the proposed in this work but offline experiments can be performed to get the accuracy with these methods.

6.9 Image detection, reidentification and tracking results

In this section all the performed experiments using the complete set of algorithms presented and the proposed new solution are collected. To this purpose, the experiments have been divided firstly by camera type applying all the detection and tracking algorithms with the

proposed camera undistortions or modifications. Secondly re-identification algorithms have been tested in two main situations, re-identification in the same camera framework and re-identification between several cameras at the same time. Finally, the complete framework with all the parts working at the same time was tested.

6.9.1 Test dataset

In order to test all the solutions in terms of person detection, tracking and reidenfication an own indoor dataset have been collected and annotated. The dataset is composed by around 1 hour of synchronized measurements in different days using the three camera types proposed. The complete videos have been annotated manually with the bounding boxes, person ID and tracking ID in the NFoV, Fisheye and 360° cameras. In the case of NFoV camera the persons were annotated not only in the collected images also after pass over the undistortion algorithm. Regarding Fisheye images, the normal and the unrolled versions were annotated. Finally, for the 360° cameras, the collected image (equirectangular projection) and the cubemap were used. The main considerations to create the dataset were the following:

- All the persons that appear in the images have the bounding box, person ID and tracking ID annotations.
- Persons occluded more than a 50% are not annotated.
- Annotations that are near to the borders in the unwrapped Fisheye, equirectangular and cubemap images have been annotated extending the images in both sides with a quarter of the complete image (see example in Figure 6.42).
- The same tracking ID is considered while ocussions are less than 70%.
- Same person ID for each concrete person is considered for records taken in the same day.

Figure 6.42: Example of an equirectangular image and the extended version to avoid person cut in borders.

An annotations example from the dataset is presented in Figure 6.43 where the true person bounding box, the tracking ID and the reidentification/person ID is shown in the images.

The selected image size for each type of camera is also an important parameter in order to achieve real time not only in detections, also in the combination with next algorithms that perform tracking and reidentification. In the case of the NFoV camera and the undistorted version the selected resolution for these images as input of the networks is 1280x720 pixels. In this case the perspective of the camera is totally frontal to the persons and the room size is not so high having the view of the persons a big size. For the Fisheye camera the selected image size is 1056x1056 pixels and cover the entire room where is installed. The fisheye unwrapped version has a size of 1000x500 pixels plus, inside the algorithms, half of the image more to avoid cut persons in the borders having at the end a resolution of 1500x500 pixels. To end, the equirectangular input image has size of 1000x500 plus the added parts having 1500x500 pixels and the cubemap is refactorized deleting the top and the bottom parts of the cube (approximately the ceil and the floor of the monitored room) having with the added parts a size of 1200x400 pixels. Is interesting have big sizes in these images due to the persons appear in more reduced sizes than in the NFoV camera.

Figure 6.43: Example of test annotation in the different images used for all the camera types. *Top:* NFOV and undistorted NFOV images. *Center:* Unwrapped Fisheye and Fisheye images. *Bottom:* 360° Cubemap and 360° equirectangular images.

6.9.2 Results on Deep Learning objects detection algorithms

All the detection algorithms presented in Section 6.6 have been tested over the proposed test dataset. All the pretrained object detection algorithms trained on COCO dataset were used directly over each type of collected image in the proposed camera environment. Frame by frame detection were performed taking into account some concrete points to obtain the final algorithm accuracy which are listed below:

- The intersection over union is used as the main metric between the annotated and the predicted bounding boxes.
- Only predicted person labels have been considered.
- A predicted bounding box which don't match with other (IoU is 0%) has an error of 100%.
- If the number of predicted bounding boxes is less than the number of annotations, no matches are free of error in IoU but not in the counter of correct detections.
- If the number of predicted bounding boxes is major than the number of annotations, the IoU for these detections is 0% then 100% error and this not matches don't increase the total detections counter.

Three main objectives that have been considered to test the algorithms. Firstly, the bounding box accuracy as the mean of all the percentages obtained by the IoU of each detection in all the annotated frames. The second is the percentage of total bounding boxes detected by the algorithms regarding the total annotated ones. Finally, the average detection time by frame is obtained. In Table 6.2 all the results are collected for each object detection algorithm:

Table 6.2: Bounding box detection accuracy in the complete annotated sequences

Algorithm	Image Type	BBox Accuracy	Detection Accuracy	Average Detection Time
Faster-RCNN	NFoV (direct)	60.43%	81.29%	59.7 ms
Faster-RCNN	NFoV (undistorted)	60.67%	81.37%	59.7 ms
Faster-RCNN	Fisheye (direct)	35.62%	10.72%	57.8 ms
Faster-RCNN	Fisheye (unrolled)	53.65%	70.26%	58.5 ms
Faster-RCNN	360° (equirectangular)	56.12%	77.84%	58.3 ms
Faster-RCNN	360° (cubemap)	57.08%	78.08%	57.8 ms
Yolo v3	NFoV (direct)	54.21%	77.23%	23.1 ms
Yolo v3	NFoV (undistorted)	54.32%	77.25%	23.1 ms
Yolo v3	Fisheye (direct)	34.35%	9.76%	22.3 ms
Yolo v3	Fisheye (unrolled)	49.32%	68.51%	21.6 ms
Yolo v3	360° (equirectangular)	52.54%	75.39%	22.5 ms
Yolo v3	360° (cubemap)	53.02%	75.1%	21.8 ms
SSD	NFoV (direct)	55.74%	78.47%	27.3 ms
SSD	NFoV (undistorted)	55.75%	78.49%	27.3 ms
SSD	Fisheye (direct)	34.78%	10.21%	26.9 ms
SSD	Fisheye (unrolled)	51.12%	69.66%	27.1 ms
SSD	360° (equirectangular)	52.69%	76.35%	27.2 ms
SSD	360° (cubemap)	53.87%	76.02%	26.8 ms
RetinaNet	NFoV (direct)	58.12%	79.23%	27.5 ms
RetinaNet	NFoV (undistorted)	58.18%	79.92%	27.5 ms
RetinaNet	Fisheye (direct)	35.08%	9.98%	26.9 ms
RetinaNet	Fisheye (unrolled)	52.91%	70.31%	26.6 ms
RetinaNet	360° (equirectangular)	54.77%	78.11%	27.2 ms
RetinaNet	360° (cubemap)	55.34%	78.31%	26.8 ms
RFCN	NFoV (direct)	57.89%	78.89%	25.4 ms
RFCN	NFoV (undistorted)	57.93%	78.9%	25.4 ms
RFCN	Fisheye (direct)	35.01%	9.39%	24.9 ms
RFCN	Fisheye (unrolled)	51.76%	68.65%	24.6 ms
RFCN	360° (equirectangular)	54.82%	77.78%	25.8 ms
RFCN	360° (cubemap)	55.52%	77.54%	24.7 ms

Results need an analysis to know what is happening in each type of image used. Firstly, an analysis about the bounding box accuracy is presented. Results are very favorable having in mind that the top result from a deep learning object detector is obtained by Cascade Mask-RCNN [282] achieving a 53.3% in COCO dataset. The following graph (Figure 6.44) resume all these data grouped by detector and type of image.

Figure 6.44: Bounding box accuracy obtained by several deep learning object detector algorithms pretrained on COCO dataset.

Firstly, the best results have been obtained by the NFoV camera. This is due to some normal reasons in particular that the point of view of the camera is totally frontal to the persons like most of the images with persons on COCO dataset. In the other hand, in this camera, due to the size of room the size of the persons in the image is big and is easy to the algorithm detect persons with this size. Finally, there is a little improvement using the undistorted image on the bounding box accuracy probably due to persons near to the borders appear a little distorted but nothing very serious. Furthermore, the best object detector is Faster-RCNN followed by RetinaNet. Both networks are very accurate in person detection. The first one is, for now the best object detector but the complexity is higher than the others. The second propose an interesting loss function that focus better the learning on the objects but the network is a little simpler than Faster-RCNN giving less bounding box detection accuracy.

Secondly, regarding all the object detectors, Faster-RCNN achieve the best performance in all images types. This is due to the robustness of this object detector. The worst is YoloV3 because is a simpler algorithm and is focus not in the best accuracy but in other aspects that will be presented in the next paragraphs. RetinaNet and RFCN have very similar results having some differences depending on the image type.

Thirdly, the worst accuracy is obtained with the Fisheye image having results around a 35%. This is a very interesting case of study. Only the good detections have been computed then the accuracy is very low. This is completely due to the perspective of the image. The camera is allocated on the ceil where COCO dataset doesn't provide any image whit persons and this perspective. The second thing is that persons appear in different sizes and forms depending where they are in the room. Persons under the camera only show the head, persons far from the camera can be seen completely but depending on the part of the image in a rotated way. The algorithms fail in detect the persons in this image if they are not far from the camera and normally in the bottom.

The unwrapped image is also an interesting case of study. After the unwrapped process some part of the image is lost (a circle around the center). Another important thing is that persons appear distorted in all the parts of the image but at the bottom they are completely distorted. Regarding the accuracy obtained by all the algorithms, it is worse than the accuracy obtained with the images from the 360 camera due to the projection don't thread in a good way the representation of the persons but the accuracy is not so bad in comparison with the other image types.

Finally, the equirectangular and the cubemap presents similar results but better than with the unwrapped image. This is probably due to the person are located at the center of the image where distortions are not so important in contrast with the unwrapped image which have distortions in all the parts of the image. Furthermore, the results with the cubemap are a little better probably due to the elimination of the distortions in the image having a NFoV image composed by the faces of the cube. The difference in accuracy with respect the NFoV image es the size and quantity of the persons that appears in the sequences having more cases with persons far from the camera in this situation.

Figure 6.45: Detection accuracy obtained by several deep learning object detector algorithms pretrained on COCO dataset.

Regarding detection accuracy (Figure 6.45) it is important to note the premises taken into account to calculate this value presented at the beginning of this section. Results are very promising having a similar behavior as with the bounding box accuracy. The best results have been obtained using the Faster-RCNN followed by RetinaNet having values of 81.37% and 79.92% respectively for the undistorted NFoV image. This high accuracy is due to most of the bounding boxes annotated have been detected by these algorithms most of the time. The decreasing from the 100% is due to some occlusions are heavily complex to the algorithms and only one bounding box is predicted instead of two.

In general, having apart the NFoV images, there are better performance in the Cubemap image having a little better accuracy than with the Equirectangular. The unwrapped image is heavily penalized due to the important distortions failing the algorithms in some detections. Finally, the fisheye image is the worst having accuracy around only 10% of correct detections. This happens due to the nature of the image where persons are totally distorted or appear in different sizes. All the algorithms fail in this image to detect persons due to no images of persons with these appearances are included in the datasets and the algorithms don't know how to detect these new perspectives. It is necessary retrain the algorithms with images taken

directly from these types of cameras.

Figure 6.46: Processing time obtained by several deep learning object detector algorithms pretrained on COCO dataset.

Finally, in contrast with the detection and bounding box accuracy, the results on processing time are different (Figure 6.46). The best algorithm is Yolo V3 due to the simple network that is having the worst accuracy but the best performance achieving real time with images of big sizes. In contrast, Faster-RCNN has the worst performance on time due to the complexity of the network but having the best accuracy in all the experiments. The rest of the algorithms also achieve real time in processing time having only between 3-6 milliseconds more than Yolo V3.

Regarding all the values obtained and looking the best performance in order of time processing Yolo V3 have been selected as the base algorithm for the complete tracking framework. The main reasons are the higher detection accuracy and the processing time. In order to complete the chain of detection, tracking and reidentification each part need time to process the previous estimations then Yolo V3 is a good selection to achieve real time processing. Bounding box detection accuracy is the worst in comparison with the other algorithms but only one point will be use as the main point to estimate the final 2D position in the room. An example of

detections using this algorithm in all the image types is shown in Figure 6.47.

Figure 6.47: Example of detections made by Yolo v3 algorithms in different image types.

6.9.3 Results on Deep Learning reidentification algorithms

This section collects all the accuracy results after train and test all the reidentification algorithms in different reidentification datasets combinations. The main idea is to include all the possible situations combining images from all the datasets where different situations appear like more complex images with small occlusions, complex lighting conditions, and different camera views and perspectives. From all the datasets presented in Section 6.31 Market 1501, MSMT17, DukeMTMC and CUHK03 were used. The main considerations to test the algorithms are the following:

- The accuracy have been measured as the RANK1, RANK5, RANK10 and RANK20 accuracy if the person is the correct with the best probability.
- The test persons provided by each dataset are divided in a query part and a database part for matching.
- Euclidean distance is used as the metric to get the most similar extracted features between persons.

- The database used for each query image don't contain the same person with the same tracking ID.

After train the algorithms in all the combinations, the test dataset presented in this section have been used for test the accuracy in the same way as the object detection algorithms. For the dataset created in the proposed indoor environment a query person has been extracted from persons with the same reidentification ID. The rest have been used to perform the matching process and obtain the accuracy in different RANKS. The Rank measurements is the percentage of success in have at least one correct person inside the first N most similar samples. As example, RANK5 means have at least one image of the correct reidentified person between the 5 most similar.

Results are very promising showing high accuracy in all the datasets. Market 1501 is an easiest dataset due to the lighting conditions are very favorable and cut persons are good defined. MLFN achieve 97.7% in this dataset but having always the best performance in comparison with the other algorithms in all the combinations of datasets.

MUDeep has the worst performance in all the experiments but having not so bad results in the RANK20 measurement achieving at least more than the 50% of successful. PCB shows in general better results than HACNN having near values to MLFN results.

The combination of datasets in general for all the algorithms has an impact in accuracy decreasing accordingly depending on the combination. For example, the combination of Market 1501 and CUHK03 present an accuracy between 80% and 87% around 10% less than with only Market 1501. This is a symptom of more complex images are in the new dataset due to the inclusion of CUHK03 dataset and with more images is more complex perform the separation with the same percentage of successful.

The same happens with the rest of the algorithms and more complex datasets combined show worst accuracy. But these are not bad news, the accuracy for the last cases with the combination of three datasets shows an accuracy in RANK1 for MLFN algorithm of 43.6%. Then the algorithm is able to get this percentage of success in select one correct person to associate in the first attempt. This combination probably is more robust due to the inclusion of data of different natures where Market 1501 contribute with more simpler cases, CUHK03 include several points of view, and DukeMTMC include both things in a more complex environment.

Table 6.3: Reidentification testing RANK accuracy (%) in different datasets (RANK1 / RANK5 / RANK10 / RANK20)

Algorithm / Dataset	MUDeep	HACNN	PCB	MLFN
Market 1501	58.9/79.8/86.3/90.4	84.5/93.4/96/97.4	83.3/92.5/95.2/96.8	87.9/95.1/96.9/97.7
DukeMTMC	48.6/67.5/73.7/79.8	71.5/84.3/88.9/91.3	72.6/85.1/89.2/91.6	80.7/89.8/93/94.6
MSMT17	16.3/29.7/37.3/46.3	33.4/50.4/58.4/66.8	34.7/52.8/61.9/67.2	36.8/55.1/64.3/68.7
CUHK03	14.4/28.5/37.8/47.2	29.1/50.4/60.5/71.7	47.6/67.7/76.9/83.8	54.9/74.4/82.6/88.4
Market 1501 + CUHK03	56/71.5/76.5/80.3	61.3/73.6/81.8/85.4	64.3/74.8/82.3/86.4	64.4/75.8/83.2/86.5
Market 1501 + MSMT17	51.5/67.5/71.6/75.3	54.2/69.3/73.9/77.4	56.5/71.8/75.9/76.7	58.1/71.9/76.4/78.5
Market 1501 + DukeMTMC	58.4/68.5/73.2/76.9	59.5/70/75.1/79.4	61.3/72.8/78.2/82.1	60.5/72.8/78.4/84.1
DukeMTMC + CUHK03	42.1/54.1/60.2/65.1	45.1/55.8/62.4/66.1	49.2/59.3/66.1/70.2	52.3/61.2/69.4/72.3
DukeMTMC + MSMT17	43.2/56.4/59.3/64.2	45.2/58.2/61.3/66.8	49.8/61.1/66.2/68.9	51.4/63.2/68.1/70.2
CUHK03 + MSMT17	40.2/48.3/53/56.2	42.2/49.3/55.6/58.1	47.3/55.3/60.1/63.3	48.6/55.3/62.4/65.2
Market 1501 + CUHK03 + DukeMTMC	38.3/42.1/48/52.7	40.8/45.1/48.1/52.6	43.1/46.7/50.2/54.3	43.6/47.8/50.7/54.9
Processing Time	12.7 ms	18.7 ms	7.9 ms	8.3 ms

Regarding processing time PCB achieve the best performance with only an average of 7.9 ms due to a simpler network which work by little convolutional blocks. HACNN has the worst mean processing time because is composed by a more complex network with several parallel branches. MLFN combine several simple blocks and has the second position in mean processing time.

The extracted features using the last combination of datasets will be very rich in order to contemplate the more cases possible to differentiate between persons. Furthermore, the pro-

cessing time is very reduced which is a good point to select the network that achieve the best RANK accuracy in all the datasets. All the networks trained in the last combination of datasets have been used over the proposed test dataset with images of all types except circular Fisheye to check the accuracy obtained in different RANKS (Table 6.4).

Table 6.4: Reidentification testing RANK accuracy (%) in different the proposed test dataset (RANK1 / RANK5 / RANK10 / RANK20)

Algorithm / Dataset	MUDeep	HACNN	PCB	MLFN
Proposed Test Dataset	45.3/49.5/54.7/60.3	51.6/57.7/64.3/71.5	55.3/60.2/68/75.6	60.3/64.8/71.5/79.8

Figure 6.48: Example of query images and RANK3 matches for different persons.

Results show that again MLFN is performing better than the other reidentification algorithms. It is achieving a 60.3% of RANK1 accuracy that is a very good result regarding the complexity of match persons in different camera perspectives, with different distortions and important differences in lighting conditions. With RANK20 the accuracy increase to a value of 79.8% then normally a 20% of the persons will be miss-classified and the twenty persons inside the RANK20 need a process of optimization in order to filter properly the correct person base on other parameters like the time, the room where the person is and the probability to go

to one or other part of the complete environment. Applying all these conditions the RANK accuracy can be improved due to not all the persons are walking all the time, only a few of them will be monitored at each moment.

An example of some query images, and the extracted RANK3 from the database is presented in Figure 6.48 where some successful detections are collected for the test dataset proposed.

6.9.4 Results on Visual (no learning)/Deep Learning tracking algorithms

Following the ideas presented in the previous experiments, all the proposed tracking algorithms have been tested in the test dataset. Based on the results obtained by the object detectors to test the tracking algorithms all the camera types annotations have been merged in a common dataset of tracks except the circular Fisheye image. The steps to test the algorithms are collected in the next statements:

- The annotated tracking id have been used for test the sort tracking person periods.
- Only one person is considered at a time to measure the accuracy.
- Bounding box accuracy measured as IoU between annotated and predicted bounding boxes is used.
- Bounding boxes with 0% IoU have an error of 100%.
- Bounding boxes that don't match annotated ones have 100%.
- All tracking algorithms except SORT and the more complex algorithms which combine several ideas in a common framework have been tested taking as first detection the first annotated frame in the annotated trajectory.
- The Time to Live for all the tracking algorithms is 30 frames and for learning algorithms a probability less than 40%.

The results are very interesting to analyze. The first point is the bounding box accuracy. It is observed that the accuracy is less than with the detection algorithms. This is due to the tracking algorithm try to estimate the next bounding box base on the data available at frame level. This estimation is not accurate as with the object detectors but the decrease

is no so important in comparison with the object detectors. Best results are achieved using Re3 tracking algorithm. The way of this algorithm works is very interesting and is able to focus better the attention on the object under tracking. No learning algorithms (CSRT, KCF, MOSSE and SORT) don't achieve better accuracy than learning algorithms showing the later an improvement.

Table 6.5: Tracking accuracy on test dataset with different tracking algorithms

Metrics / Algorithms	Mean BBox accuracy	Mean Detection accuracy	Mean Processing time
CSRT	36.2%	74.6%	20.1 ms
KCF	43.5%	81.6%	10.2 ms
MOSSE	45.3%	82.7%	34.4 ms
SORT	41.6%	79.4%	15.8 ms
GOTURN	44.8%	82.7%	25.7 ms
Siamese Net	46.9%	84.4%	27.4 ms
Re3	47.8%	85%	15.6 ms
Deep-sort	43.5%	82.2%	40.2 ms
Det-track	52.6%	84.7%	120.1 ms
Alpha-Pose	53.5%	85.1%	62.4 ms

In the other hand, other algorithms which include both, detection and tracking present a very good behavior in the tracking task. Det-Track and Alpha-Pose achieve an accuracy close similar to the accuracy obtained with Yolo v3 object detector. In the case of Alpha-Pose, the network use directly Yolo v3 before the matching process for tracking. Det-Track use RFCN as base for object detection and the accuracy is close similar to the accuracy presented in previous experiments over the proposed dataset.

Regarding detection accuracy it is important to note that the accuracy is in general better than with the best object detector (Faster-RCNN). The reason is that object detectors fails some times in the detection in some frames, the tracking solve the problem having a continuity during the frames, then, increasing the number of frames with detections. Re3 is again the best tracker achieving 85% of detection accuracy. In contrast Alpha-pose is more robust due to is focus only in persons and achieve a little better accuracy in the complete combination

of person detections a tracking matching.

Deep learning algorithms achieve better performance than computer vision trackers. This is an important notice breaking the computer vision traditional ideas with the learning algorithms. Finally, mean processing time is analyzed. The best result is obtained with KCF. This tracker is very simple and efficient but is far from the bounding box accuracy obtained with Re3. Re3 is the next one having worst performance in processing time due to is very sensitive to the number of trackers in the image. For only one tracker is able to achieve 100 FPS but if the number increase this time increase dramatically. Furthermore, Det-Track and Alpha pose have an elevated processing time making impossible a real time processing for some cameras working at 25-30 FPS.

Due to the learning algorithms need the initial bounding box to track, detection part has to be performed all the time in order to find new persons and a new stage is necessary to delete detections that are being tracked before. This problem can affect the final performance of a complete solution. Then learning algorithms have very good accuracy but lack in multi-tracking ideas and one network is necessary by each detected person avoiding the real time processing with more than two persons at the same time being tracked.

Computer vision algorithms also lack in multi-tracking but there is no so serious track several persons at the same time with these trackers. The initial bounding box should be given then present a similar problem like learning algorithms needing launching one tracker per detected person. Depending on the number of persons this solution can be not the best increasing heavily the processing time with more than around ten persons in the image being tracked.

Finally, SORT algorithm has been selected as the base tracking algorithm. It is not the best one and only use bounding box information for next state predictions with a Kalman Filter. No image data is used in this tracker. The main advantage is that the inclusion of more and more trackers to be tracked is not a serious problem due to the ability of the algorithm to work with matrices performing optimal computations with a non-important increasing in processing time.

Figure 6.49: Example of tracking occlusions and tracking ID change in three types of images.

6.10 Proposed framework for Detection, Tracking and Reidentification (DTR)

To close this section the complete algorithm developed using a pretrained object detector (Yolo v3), probabilistic tracker (SORT) and a trained reidentification algorithm (MLFN) in a dataset combination is presented. Two reidentification steps appear in the complete algorithm. The first one tries to reidentify persons in the same camera where occlusions or tracker losses appears. The second one performs the reidentification between different cameras to match persons that go between one part of the monitored environment to others.

All the cameras are connected to the same network where the system is deployed via wired or wireless connection. All the images are sent or collected directly in the main server that perform all the steps involved in the complete framework. The first step is the detection process that comprises two main tasks. The first task is to apply the object detection algorithm selected after testing to the images collected by the cameras. Due to the different frame rates

images are available at different timestamps but if more than one image is available the GPU is capable of process more than one image at a time due to the high capacity.

Once the image is processed, timestamp and person detection accuracy are evaluated to only maintain detections with more than 50% of confidence. These persons are cut from the image and pass directly over the reidentification algorithm after a resize to adapt the network input size in order to extract a feature vector per person detected in each frame.

Next step is to apply the tracking algorithm selected in order to match previous persons detected with the new ones. A tracking ID is assigned to each person. Tracker flow works in the following way:

- New trackers are added to the current list of followed trackers.
- Trackers losses pass directly to the first reidentification step (internal same camera matching).
- Trackers are updated if all the conditions have been sufficient to match previous detection with the new ones.
- Trackers have a Time to Live in order to finish the tracking if no new detections can be matched with previous estimations.

Once one tracker is lost goes to the first reidentification step, the in-camera matching. Firstly, the mean of all features vectors extracted frame by frame is taken as the final feature vector for comparison. This is a more robust feature vector for comparison due to take into account all the possible changes in the point of view of the person during tracking along several frames. One important parameter here is the time-to-live of the tracker. Trackers are considered for matching until this time pass.

Figure 6.50: Complete workflow of the final proposed framework for detection, tracking and reidentification.

The first reidentification step use the extracted mean feature vector to perform the comparison between persons in the same camera. To do this Munkres algorithm is used to match the feature vector with previous lost trackers if the Euclidean distance is below than a threshold. Some considerations are involved in this process. The first one is which threshold should be used. In order to include the most information possible the threshold after several experiments is set to a concrete value based on the distances extracted from the test dataset in persons that match in the same camera. After this process the best match is associated as previous tracked person like the same person with same reidentification ID. Another important parameter is the time between cameras. This is an important value to only consider that a new tracker can appear in a different camera after a concrete time that you need to go walking between cameras.

Finally, the last step perform a close similar process as in the first reidentification step. In this case the matching between cameras that don't cross the point of view is done in order to match persons across the entire environment. Some thresholds have been defined like the minimum time to go between two cameras (avoiding the matching process of a person until this time ends) and the distance threshold to consider one feature vector similar to other.

After all the presented steps all the bounding boxes for each reidentified person are obtained. To end the process, pixel coordinates must be converted to 3D coordinate space to match the image environment into the same scenario used with the wireless sensors. This process needs have all the cameras calibrated to get the camera intrinsic matrix and the distortion coefficients. For the 360° camera the process to can calibrate is convert the image into double fisheye and calibrate both lens as two independent fisheye cameras. In this way a point can be projected from the pixel space to the world space.

The same videos have been used to check the accuracy of reidentification algorithm in both cases into the same camera matching and between cameras. The accuracy of the object detector and the tracking algorithm have been tested in the previous section then only the accuracy in the conversion between pixels and world coordinates can be tested. Due to the impossibility of annotate perfectly the proposed test datasets some sequences with predefined trajectories have been collected having a total of 3 minutes per camera and annotated manually with both, pixel and real coordinates. The idea of these trajectories is trying to cover all the parts in the image to check that the conversion works in most of the parts of the image.

Table 6.6: Reidentification accuracy and World coordinates conversion accuracy

Algorithm Part	Accuracy
In-Camera Reidentification NFoV	92.4%
In-Camera Reidentification Fisheye	88.8%
In-Camera Reidentification 360°	96.4%
Reidentification between cameras	85.4%
NFoV camera	49.3 cm
Fisheye camera	42.2 cm
360° camera	43.5 cm

Table 6.6 collects all the final results extracted from the test dataset and the collected sequences to test the conversion accuracy. Firstly, the in-camera reidentification accuracy have been tested. The results for each camera are very promising. In the case of the NFoV camera this value is due to some complex occlusions and tracking failures during the sequences. Regarding, Fisheye camera, the accuracy is worse than with the other cameras. This is due to the important deformations that happens in the image where a person is being tracked an appear in different parts of the image with different sizes but the adoption of the mean feature vector to characterize the reidentification vector helps in this case. For the 360° camera the accuracy with the Equirectangular image is the best and is due to the trackers are more easily tracked and appear during longer periods having a more robust feature vector for each one.

Finally, the accuracy obtained after the conversion to real world coordinates the mean accuracy in the collected sequences is very promising having an accuracy in all the cases of less than 50 cm. It is important to note that cameras are more reliable than sensors and wireless signals to perform these tasks. The mean accuracy for the Fisheye camera is the best probably due to two main factors, a good calibration and a easier point of view to select the marks for calibration in the floor. The NFoV conversion accuracy is the worst due to the complexity of the environment with several furniture in the middle of the image increasing the complexity to select proper points to calibrate.

6.11 Research contributions for indoor tracking with cameras of different field of view

Lots of experiments have been performed using different types of cameras and different image projections to get the detection, tracking and reidentification accuracy in indoor environments.

Initially, concepts about image projections and camera calibration have been presented in order to work with different types of images coming from the same camera and to know how to convert between pixel coordinates to real world coordinates. Some of these ideas are presented in the paper [283].

Regarding detection, deep learning approaches have been tested having good results in more challenging conditions like low light, people occlusions and different camera points of view.

In the case of tracking algorithms, a comparative between computer vision and deep learning approaches have been presented.

Finally reidentification algorithms to extract rich features to identify univocally a person have been used in order to create a final complete architecture to support efficiently the tasks of detection, tracking and reidentification of persons.

Chapter 7

Multisensory IPS Fusion

7.1 Person Identification and Multi-modal Fusion

In order to combine the information from the sensors, a previous *identification* stage must be performed. The reason is that multiple factors such as target occlusions, lighting changes, signal attenuation, reflections, among other effects, affect the tracking algorithms yielding to estimation errors. In fact, for scenarios such as the ones considered in this work, affluence of several persons is assumed and therefore a classification tool to associate the routes estimated with the corresponding person is required. Furthermore, the identification task must be performed for each sensor which implies that several matching levels from sensors data are needed.

Two main inference levels have been proposed:

- *Track-to track comparison*: using the trajectories extracted from the wireless signals and sensors, and the trajectories extracted from each deployed camera.
- *Orientation comparison*: Using the heading estimation from accelerometer, gyroscope and magnetometer and the estimated orientation from camera trajectories as direct geometry angle calculation.

Because of the system modularity, the identification process must be performed even if there is

no data gathered from all sensors due to occlusions or coverage. Conversely, if measurements from sensor devices, the three types of cameras are available, the two comparison methods can be applied to associate the estimated trajectories to a person in the scene.

7.2 Data synchronization

When the collector system gets the sensor information, store this data with a timestamp. Each device (sensors and cameras) has his own data rate and theses timestamps are different for all measurements. The sensors data rate can be shown in the next table:

Table 7.1: Data rates from the devices involved in the system

Sensor	Data rate (measurements per second)
Wireless sensors device	30
NFoV camera	25
Fisheye camera	15
360° camera	30

The synchronization task need takes all this information with different timestamps and fix time interval where the measures are considered. The problem to solve is the variant data rate. The next graph shows an example of the situation:

Figure 7.1: Example of data collected in time with different data rates.

In Figure 7.1 can be seen several balls with colors. Each color represents one sensor and each ball represent a measure from this sensor. Each ball has its own timestamp. To synchronize these measurements, we need to control the time. For this purpose, it is defined a windows time where measurements inside this window will be synchronized. The next graph (Figure 7.2) shows an example of this concept:

Figure 7.2: Sensors data inside predefined windows time.

To take the value for the duration of this window we need to see the data rates. To achieve get measures from all sensors in each window and avoid windows with blank space of one or more sensors this relation is used:

$$\tau < \frac{1}{\min(data_rate)} \text{ st } \tau \leq threshold \quad (7.1)$$

This formula takes as windows time the sensor with minimum data rate to guarantee almost one measure per sensor in the window if the sensor is available. Once the windows time is selected the synchronization process takes place. This process merges the measurements from the same sensor in only one for all available sensors in each windows time. For this fusion it is use the mean of the measurements but other metrics like the median can be used. The threshold parameter in the formula is one value used to guarantee the stability of the measurements after the mean performed over the measurements. If the windows time is bigger and for good estimations it is needed a lot of measurements to see little changes between them, this requirement can be lost. To select this value, it is needed perform some evaluations before and after applied this method and see if the precision loss is no so important. After this process the final output can be seen in the next graph (Figure 7.3):

Figure 7.3: Data after synchronization.

Now for each window we have almost one measurement per sensor available and for set up the final timestamp it is used the half timestamp for each window.

7.3 Track-to-Track and Orientation comparison

Assuming that data gathered have been synchronized in the previous stages of the system using the proposed approach, the goal of the Track-to-Track procedure is to detect the routes with the higher similarity. The metric employed is the Minimum Square Error MSE along the route duration and can be expressed as follows:

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to} \quad f_i(\mathbf{x}) \leq t \quad i = 1, \dots, m. \end{aligned} \quad (7.2)$$

$$f_i(\mathbf{x}) = \frac{1}{M} \sum_{j=1}^M \sqrt{(x_j^i - x_j^j)^2 + (y_j^i - y_j^j)^2} \quad (7.3)$$

where x, y are the axis coordinates of a predefined route, m represents the number of routes in a scene (frame or sensors time interval). M is the number of trajectory points taken into account in the estimation (corresponds to the time interval where synchronization is performed). The cost Matrix can be created with the MSE combinations between the mentioned sensors. As

an example, if there exist two sensor devices having two trajectories and Fisheye camera is tracking three persons, the cost matrix is determined as:

$$C = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \end{bmatrix} \quad (7.4)$$

where every t_{ij} is obtained by solving the optimization problem (7.2). Munkres algorithm [284] is applied to the matrix (7.4). If all conditions are satisfied, the final assignment can be carried out and therefore link the trajectories for a determined person.

As an example, Figure 7.4 shows several trajectories in different time moments from all the sensors involved. The main idea is try to match trajectories that are close similar and are independent in time. Normally the sensor devices will be tracked all the time but cameras will have short tracking period where persons are visible.

Figure 7.4: Example of trajectories from all the devices involved in different time moments.

Orientation comparison is proposed as second level of inference for person identification. Sensor devices are equipped with sensors that directly provide the heading orientation. The comparison will be performed using the metric described in (7.2). The filtering process of the obtained curves is then executed to obtain the MSE. Analogously to the previous inference level, Hungarian algorithm is employed for the assignation process. An example of the yaw obtained directly from the sensors device and from one trajectory extracted from the Fisheye camera and its filtered version is presented in Figure 7.5.

Figure 7.5: Example of a part of a trajectory yaw from the Sensors Device and the Fish-eye camera for the same person.

7.4 Multimodal trajectory fusion

Once the identification procedure is performed, the measures can be fused in a multi-modal scheme to improve the accuracy of estimations. A fusion algorithm can be used to achieve a high level of accuracy estimation using the information from cameras and WSN. Kalman Sensor Group Fusion algorithm (KSGF) is employed to perform the final fusion. This algorithm exploits the Kalman filter properties for multi-sensor data fusion estimating a final x, y coordinates based on two trajectories. This is due to trajectories from cameras have been previously associated using the reidentification IDs and the other trajectory involved in the

system is the extracted sensors trajectories. The filter has the following form:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (7.5)$$

$\mathbf{H}$ is the state matrix for Kalman filter. Furthermore, $\mathbf{R}$ matrix includes sensor error covariance measurements. These values are appended to the main diagonal:

$$\mathbf{R} = \begin{bmatrix} \sigma_{x_{sensors}}^2 & 0 & 0 & 0 \\ 0 & \sigma_{y_{sensors}}^2 & 0 & 0 \\ 0 & 0 & \sigma_{x_{cam}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{y_{cam}}^2 \end{bmatrix} \quad (7.6)$$

Finally, the observations vector $\mathbf{z}$ collects the x and y coordinates estimated from each sensor:

$$\mathbf{z} = [\hat{x}_{sensors}, \hat{y}_{sensors}, \hat{x}_{cam}, \hat{y}_{cam}] \quad (7.7)$$

As an output of the KSFG stage, a trajectory fusion estimation is obtained. This is the final goal of the complete project. To test this final step the 1 hour recordings used in the cameras experiments have been used where one person had the device attached to the hip during all the recordings. The idea is to get the accuracy with the wireless sensor device, the accuracy of the trajectories of the persons in the monitored environment with the three camera types and the final accuracy obtained with the fusion including the previous association step.

Table 7.2: Final Mean Error Accuracy for each sensor and after the fusion process in the proposed test dataset.

	WSN	NFoV Camera	Fisheye Camera	360° Camera	KSGF
Accuracy	54.41 cm	49.3 cm	42.2 cm	43.5 cm	40.32 cm

Table 7.2 shows the accuracy obtained in the previous sections for the developed sensors system with the second configuration (Swiblux) and the accuracy obtained in the trajectories

extracted from each camera. Thanks to the high accuracy of the two level reidentification (in-camera and between cameras) most of the trajectories are correctly associated to the correct person ID in the camera system. After this process the fusion takes place firstly associating the camera trajectories with the sensors trajectories and secondly fusing the information.

It is observed in the final results a little improvement with respect the mean error accuracy extracted from the sensors independently. Sensors can compensate errors in the camera trajectories due to a bad conversion from pixels to real world coordinates. In the other hand the same happens with cameras where better accuracy fused with sensors trajectories improve the results obtained for track persons with both systems.

7.5 Research contributions in the final fusion of trajectories extracted from different systems

A complete synchronization and fusion architecture have been presented in order to fuse the information which come from the sensors system and the information from the cameras system. An increasing in the final accuracy of the tracking system is observed and the module is able to associate camera trajectories to sensor trajectories which are the same persons.

The system can be used with different systems where the output is the estimated trajectory and the fusion algorithm is able to combine lots of trajectories fixing properly the parameters to the fusion. Some own works that support these ideas are [283] and [285].

Chapter 8

Real applications using the proposed system

To conclude, this section is intended to show that the proposed modular framework for wireless tracking with sensors and accurate tracking with cameras can be used in several scenarios where some of these sensors are available. Firstly, an application related to healthcare using a combination of several bracelets with wireless signals and motion sensors and a set of cameras deployed can monitor the patient's movement in homes and day care center. As a second approach, the image base part presented in Chapter 6 is applied without modifications but in a totally different domain in order to work with faces in video.

8.1 Patient tracking and monitoring for local and remote rehabilitation at home

Elderly people care is actually a big challenge and a very interesting field for the research community. This represents a valuable opportunity to develop scalable applications to cover the special needs in terms of health monitoring and accessibility for people with cognitive impairments. In solution, a complete system to support daily activities of elderly people based on a multi-sensor scheme have been developed but in this work the parts that take the advantages of the proposed complete framework is presented. This system is intended to be

deployed not only at home, but also at crowded places such as daily care centers.

The system is able to detect several persons simultaneously and associate the data from the sensors considered to an individual. This is a significant advantage of the proposed system as it allows to extend the scope of detection and tracking to multiple patients. Several works have been presented for integrated healthcare using ICT-based solutions. A complete review of WSN systems for healthcare applications is drawn in [286]. Moreover, traditional health monitoring systems rely on the use of vision sensors [287]. Recently, implementation of depth sensors have permitted to analyze body movements in a detailed manner by the employment of interesting features such as the joint skeleton detection [288].

8.1.1 Proposed solution for Patients monitoring

A complete scenario is composed by several sensors. Firstly, Kinect camera V2 is a multi-sensor device developed by Microsoft. It is able to provide up to 30 frames per seconds (fps) getting the RGB, depth and skeleton joints in the 3D space. Both images and skeleton joint points are labeled with the corresponding timestamps. A Fisheye Camera is employed due to the use of special lens that can increase the range of detection which represents an important advantage for large range applications. The configuration is the same as the proposed in previous sections along this complete work. The Wireless Sensor Network (WSN) consists of a group of self-organized sensing nodes deployed across the area to be monitored with same configuration as the used in this work. Finally, patients wear the proposed sensor device attached to the hip. Figure 8.1 shows the deployment of the sensors in a patient home.

The house is divided in four main rooms. The bedroom has a bed, some furniture like shelves, nightstands and lamps simulating a real scenario. The living room is equipped with a Smart TV, a table, some chairs and a static bicycle. The bathroom has the main elements necessary in this room. To finish, exists a complete equipped kitchen with all the electrical appliances like a fridge, oven, sink and washing machine. This configuration is ideal to simulate a real indoor scenario for the system.

The devices were sitting in a location that given the maximum useful information possible. Two Kinect cameras were deployed, one in the Living room that covered the complete space containing the chairs and the Smart TV. The second one was sitting in the Kitchen covering all this room to detect the maximum interaction of the patient with the elements in this room. The Fisheye camera was located in the center of the house on the ceiling covering the entire

space. WSN nodes were located at the corners of the house covering all possible spaces.

Figure 8.1: Deployed system in a home composed by a WSN, a Fisheye camera and two Kinect cameras.

With this configuration of the system all the important parts were visible and all sensors can work at the same time collecting data from activities, important events and patient's movements.

Wireless Sensor Network provide directly the trajectory of the person monitored using the proposed system in Chapter 5 using the measurements taken from the device attached to the person. Kinect camera provide the skeleton joints. Using a stable point like the neck or the chest point the trajectory can be extracted directly in the 3D space. From Zenith camera, the unwrapped image is used to extract the final trajectory from the calibrated camera in the room coordinate system. Examples of images taken from Kinect cameras and Fisheye camera are provided in Figure 8.2

Figure 8.2: Example of images taken by the deployed system in a home. *Left:* Image from the Kinect camera installed in the living room. *Right:* Image collected by the Fisheye camera mounted at the ceiling.

8.1.2 Results of the system in a real scenario with patients.

To summarize, the experiments were performed in several sessions with a total of 18 available patients being monitored in a controlled and calibrated area similar to a real home scenario. The control group has been composed of elderly people without cognitive problems. Additionally, the experimental group was formed by patients that were classified into three groups according to the illness stage as: initial, moderate or severe. Diverse aspects concerning gender balance as well as related diseases have been considered.

The experiments presented is intended to show the identification and individualization process. Numerical results of the activities detected by the sensors separately are drawn in Table 8.1. Patients were recorded while walking for around 3 minutes. Additionally, the improvements reached by the multi-modal fusion are detailed.

Regarding obtained results from the dataset collected with the patients some interesting conclusions can be extracted. Firstly, object detection algorithm and Tracking algorithm perform worse than in the proposed environment in this work. This probably due to a more complex distribution of the room to monitor efficiently the patients using the fisheye camera. In contrast the reidentification accuracy is for RANK5 the 100%. This is due to in the experiments performed and the dataset collected, patients have different clothes very easy to identify and also the number of persons involved is resourced in contrast with the experiments

Table 8.1: Detection Accuracy, Tracking accuracy and Person re-identification success percentage based on the number of person presented in the monitored room at the same time.

Algorithm	BBox accuracy	Det accuracy	Mean processing time
Yolo v3 (Fisheye Unwrapped)	54.9%	61.6%	21.6 ms
SORT	49.2%	65.1%	10.2 ms
Reidentification accuracy	RANK1	RANK5	RANK10
Re-Id Fisheye Unwrapped	95.7%	100%	100%
Algorithm	Mean tracking error		
2D Fisheye unwrapped			104.78 cm
Kinect tracking accuracy			40.58 cm
WSN tracking accuracy (Swiblux)			105.62 cm
Sensors		Final fusion accuracy	
WSN + Kinect + Fisheye			80.56 cm

performed in the proposed indoor scenario in this work.

The tracking accuracy after the conversion of all the sensors involved to the same coordinate system shows very good results. For the Zenith camera the mean error is around 1 meter. Kinect camera is very accurate but the problem is the detection range that is able to cover, around 6 meters frontally. The Wireless Sensor Network also shows worst results but this is directly due to not a good calibration as the performed in the proposed environment have been done or the complexity of this environment is heavy to get better accuracy. Finally, the fusion achieve a mean error of around 80 centimeters. This is a very good result where all the sensors are involved and the influence of Kinect camera achieve reduce the mean error with all the sensors at the same time.

It is important to note which is the application in this scope. It is not necessary a precise indoor localization for the patients, it is necessary only know in which room the patient is along the time. Having this information, it is possible to know how many time the patients go to the bathroom, how many time pass for example sleeping or in a dangerous situation where a patient can be confused know if the patient is leaving the house to launch an alarm to advice the problem to caregivers.

In Figure 8.3, the location of the patient in the home is shown for a day for a real experiment with the system deployed in one home. The living room area was the area covered by the whole set of sensors. The detection was mainly obtained through the WSN (via the sensor device). The accuracy of the detection is high as the resolution is not so large (a room).

Figure 8.3: Evolution of patient location within home over a 24-hour period. The time the patient is outside is not monitored by the system.

As a conclusion, it has been demonstrated that the multi-modal approach allows to improve the accuracy of locations estimated which is of special interest for monitoring of patients with disabilities or mental impairments. Additionally, WSN in combination with the trajectories can be used to detect anomalies such as the patients leaving the house. All this work is supported by [285, 289, 290].

8.2 Automatic detection and tracking of characters in the scene for accessibility services

Universal access on equal terms to audio visual content is a key point for the full inclusion of people with disabilities in activities of daily life. As a real challenge for the current Information Society, it has been detected but not achieved in an efficient way, due to the fact that current access solutions are mainly based in the traditional television standard and other not automated high cost solutions. The arrival of new technologies within the hybrid tele-

vision environment together with the application of different artificial intelligence techniques over the content will assure the deployment of innovative solutions for enhancing the user experience for all. In this work a set of different tools for image enhancement based on the combination between deep learning and computer vision algorithms is presented. These tools will provide automatic descriptive information of the media content based on face detection for magnification and character identification. The fusion of this information will be finally used to provide a customizable description of the visual information with the aim of improving the accessibility level of the content, allowing an efficient and reduced cost solution for all.

8.2.1 Proposed accessibility services for the TV

Firstly, a face magnification service. This tool is aimed to allow the end-user to better access specific areas of the content such as faces for improving the intelligibility on the video. This tool can be directly activated from the specific menu in the user application. When the option “face magnification” is enabled, the application will automatically zoom in and out the video in real time, using the (x,y) coordinates from the tracked faces in the video to focus the face and the height and width to adjust the zoom scale. The app will zoom in either the character is speaking or not. If two or more faces are detected, the app will zoom the character that is speaking, while if no one is speaking or more than one is speaking the app will zoom out to show all faces.

Figure 8.4: Example of automatic face magnification in a preprocessed content.

The second proposed service is the character recognition and characterization. This tool is aimed to allow the user to know which characters are on the scene at a specific moment. When this function is activated some information appears in the video: picture of the character, age and gender, and the person who is talking. If needed, the age can be presented as a number or as a defined age interval such as children, young, adult and elderly.

Figure 8.5: Example of information extracted from faces showed in the video.

8.2.2 Proposed approach for accessibility services using Deep Learning techniques

An specific scenario in which users with disabilities can consume TV content with people without disabilities on equal basis by using a combination of advance image processing methods and HbbTV [291] is considered. In this scenario a TV is connected via HbbTV to a companion screen in which the user can make use of these accessibility services. At this stage it is important to note that these solutions are defined to be applied to broadband pre-processed content (that means that the main applicability is to pre-stored content such as in Video-on-Demand), considering their application to broadcast delivered content as a possible future research work which will be fully dependant on the existing software and hardware computational capabilities.

A complete framework has been created to extract image information from the multimedia broadband delivered content to feed the two access services. The proposed architecture uses several artificial intelligence and deep learning techniques trained with existing datasets together with information from our media contents to better fit the main information retrieval techniques:

- Face detection: this part is intended to find the location of the character's faces over the video.
- Face tracking: the goal of this component is to achieve a smoothed tracking over faces for the image magnification service.
- Face recognition: the main purpose of this module is to detect and recognize different characters along the video.

- Face speaking: this part is in charge of estimating which character is speaking (this part is out of the scope of this work).

The proposed framework demonstrated that all these algorithms can work together and provide better access solutions when retraining these networks with the targeted new data. The complete architecture of this framework is presented in Figure 8.6.

Figure 8.6: Complete framework for face detection, tracking, recognition and characterization.

The framework takes the advantage of the architecture presented in Chapter 6. A first module performs the object detection in this case focuses in detect face than a retrained version of the object detector is needed. It includes a scene detector module for detecting the changes between scenes, which may help the face tracking along the video. After the tracking module is applied over the detections. Finally, a recognition module is used by the reidentification algorithm but in this case without remove the last part of classification using a fully connected network.

Video contents are processed frame by frame to extract information from the whole video. Detection parts are applied at frame level and tracking parts are applied at frame level but taking into account previous and next frames information. After the extraction of all the involved data another process cleans the information using temporal connections between each scene in the video.

Face Detection is the main part of the entire architecture, since if no faces are detected, no postprocessing can be done to extract related information from the video under analysis. For this reason, and in order to enhance the process, a well-known dataset such as FDDB [292] for face detection has been used for training the algorithm.

The next step is to sample the video for detecting where the scene change appears. This is an important task for characterizing each scene with the appropriate information. Moreover, it also gives essential information about when the tracking algorithm should stop and start over with a new face detection in the next scene.

Face Tracking is the next step and provide information about where the faces are located for their magnification. If it was only based on the face detection algorithm, different important problems would appear, such as:

- Bounding box size: since it is not fixed for the face detector along the scene, the face magnification process will be noisy due to the fact that the center of this box is the one selected for performing the magnification.
- Detection loss: face detector is able to detect faces in single images but it may present different problems in complex situations, such as when the face is not pointing directly to the camera, reducing its accuracy along the scene time. Tracking over the detected faces solves this problem by maintaining where the face is located if no detection is achieved.

In another aspect, several sequences have been annotated in own videos to test the accuracy in the trajectories obtained with the tracking algorithm (SORT). A total of 100 video sequences from different contents have been used to perform the testing's.

For Face Recognition the input are the cropped faces from the previous estimations after tracking algorithm, and it uses the reidenfication MLFN algorithm to perform a classification. The output is the same that the labels in the applied network, which are the names of the main characters and one more label identified as “unknown” for others. Considering this, when one of the main characters is recognized, the presentation module will display his related information. In contrast, if some other faces are detected and labeled by the network as “unknown”, these faces pass along another branch of the network in order to estimate the age and gender to collect a general information for each scene (but this part is out of the scope of this work).

To train this network, a set of different faces of the same person in different situations have been collected and annotated. In this regard, our dataset includes more than 60.000 images for training, divided into 17 different classes (16 main characters plus 1 label for unknown) in the case of the video content presented here, and 15.000 for testing. Figure 8.7 presents an

example of some of the main characters together with a set of different images stored in the dataset of only two of them.

Figure 8.7: Examples of the main characters (top) and some images in our dataset for some of them (bottom).

Finally, all the results for each part, detection, tracking and identification have been collected in Table 8.2. For the object detector (Yolo v3), bounding box accuracy, detection accuracy and average detection time values are collected for both, training and testing and have been obtained following the same premises as in Chapter 6. Figure 8.8 shows an example of detected faces in video. To evaluate the performance of the tracking algorithm (SORT), bounding box accuracy, detection accuracy and average detection time have been obtained. Furthermore, for the MLFN identification the classification training and testing accuracy is obtained along the mean processing time.

Table 8.2: Data rates from the devices involved in the system

Algorithm	BBox accuracy (Train)	Det accuracy (Train)	Bbox accuracy (Test)	Det accuracy (Test)	Mean processing time
Yolo v3	79.8%	92.5%	77.8%	90.1%	21.9 ms
Algorithm		BBox accuracy	Detection accuracy	Mean processing time	
SORT tracking		73.8%	92.4%	8.9 ms	
Algorithm		Classification accuracy (Train)	Classification accuracy (Test)	Mean processing time	
MLFN (classification)		97.8%	92.5%	8.7 ms

Figure 8.8: Example of detected faces in video.

Results are very interesting confirming that the proposed framework for detection, tracking and reidentification can be applied in other domains training properly the algorithms in different datasets. Firstly, the detection algorithm achieves a 79.8% and a 92.5% accuracy in bounding box and detection respectively after training the algorithm. At test time the accuracy of bounding box detection only decreases to a 77.8% and the detection accuracy is 90.1% very high values having a good detector for faces. The processing time is close similar to the obtained in previous experiments due to the size is the same as with the NFoV camera.

Tracking algorithm also shows great results but with less bounding box accuracy due to deformation during the estimations when the faces are tracked or fast movements and occlusions appear during the detection. Detection accuracy is also a little lower but showing a good performance to track faces efficiently.

Figure 8.9 shows an example of the comparison of the results obtained by using face detection vs face detection plus tracking. The curves present (x,y) coordinates of the center of the detected face of both approaches, where the first one gets the coordinates of the detected face frame to frame while the second one starts with a face detection in the first frame of a scene, followed by face tracking. When using face detector, some faces are lost for some frames during a scene. This makes that the faces will be continuously reduced and magnified, giving a bad experience to the user. Regarding the face detection plus tracking algorithm, two important results can be observed: firstly, if the face is not detected by the face detector, tracking algorithm is able to continue detecting it. Secondly, the tracking curve is smoother than in the first solution. These two facts make the algorithm filter better the trajectory avoiding very noisy transitions.

Figure 8.9: Performance comparison between the face detector applied in all frames and the tracking over the first detected face.

To end this section, the reidentification algorithm retrained for classification tasks (maintaining the fully connected final layers) also work well having very high accuracy in character recognition with a total at test time of 92.5%. This task is performed every frame where the face is tracked then if some detections fails during tracking can be filtered using the most repeated values along the entire tracking of the concrete face. This work is presented in [293].

Chapter 9

Conclusions and future lines

9.1 Conclusions

Firstly, regarding the three evolution's of the tracking system using wireless signals and inertial sensors a complete framework for accurate indoor tracking has been described. A complete overview on Fingerprint construction and assessment has been presented. On the one hand, this work showed the impact of different positioning methods using RSSI Fingerprinting technique on the estimation accuracy.

The first system is composed by several algorithms that allow to adaptively track an object in an indoor scenario. Furthermore, the system takes advantage of the multiple information sources by means of a fusion strategy. It can be concluded that the fingerprinting technique in combination with probabilistic techniques such as Kernels outperform the results obtained by employing deterministic techniques. Moreover, in the view of results obtained, it can be concluded that the Track-to-Track Fusion strategy outperforms Kalman Sensor Group Fusion. Finally, in general terms, technologies employed can be fused to improve the accuracy of estimations.

The second proposal includes a custom filter proposed to improve the position estimation results and finally, a Tracking Algorithm was applied to adjust the measurements to a known movement model. The Fingerprint dataset was constructed taken into account the Fingerprint parameters for diverse situations. Important aspects such as the orientation and the human

presence were considered and it has been shown that a significant improvement is attained compared to other estimation procedures. These parameters allow to obtain a consistent feature vector for training Machine/Deep Learning algorithms. Furthermore, the inclusion of the Yaw/Heading information in the feature vector enhances ML/DL training accuracy up to 15%. Final results in Fingerprinting estimated position using learning algorithms outperform the common deterministic and probabilistic methods up to 45 – 50%.

The proposed Gaussian Outliers Filter with the weighted combination of the output probabilities yields to an increase the accuracy around 10 – 15% in Fingerprinting position estimation by avoiding impossible states or low likely estimations. Deep Learning approaches provide better results than Machine Learning algorithms when the input data is well conditioned and representative.

Particle Filter was applied to adjust the final estimations to a realistic movement model. The use of the estimated velocity and the Yaw/Heading angles outperforms the final accuracy presented by all the methods without applying a Tracking Algorithm. An improvement of 25% is achieved after using this filter over the Fingerprinting estimations.

The last proposal presents a complete tracking architecture using RNNs. On the one hand, it has been presented the attractiveness of employing such networks to learn and predict a human movement model based in some physical parameters such as position, orientation and velocity in combination with the RSSI parameter collected by a Wireless Sensor Network WSN. On the other hand, a novel proposal to create new measurements subject to the time of the day using generative models was drawn.

The Fingerprinting dataset was collected along several days to have enough information to characterize properly the environment during periods with high people flow and periods with less persons in the room. Furthermore, several labeled routes were gathered using the device equipped with the Inertial Measurement Unit (IMU) to have all the parameters involved in the proposed feature vector as input of the Recurrent Neural Network.

Diverse RNN configurations were tested showing that the use of both the proposed feature vector and the modified MSE Loss function with a regularization parameter increased the accuracy in the position estimation in relation with other combinations up to 3%. Furthermore, the comparison with other works shows and improvement in the mean error of 36.7% (reduction from 100.72 cm to 63.76 cm).

The Conditional GAN was tested comparing the mean and variance of the RSSI maps with

respect to the real RSSI maps generated. It was shown that the similarity between the behavior of the RSSI in the real maps is significant to the presented in the generated maps.

Additionally, the second RNN was able to learn efficiently from the input feature vectors to predict the position using the generated maps when no measurements are available. It reached a large accuracy in the estimations in short window times having similar results to other works in the literature.

Secondly, an image-based solution using different types of cameras and a combination of detection, tracking and reidentification algorithms has been developed. Well known algorithms have been tested for deep learning object detection with focus in person detection have been used. The effectiveness of these methods has been presented having very high accuracy person detection in all the proposed image types except in the direct Fisheye image. All the results have been analyzed adopting finally Yolo v3 due to the obtained accuracy and the processing time.

Different computer vision and deep learning tracking algorithms have been also tested in order to track the detection's and individualize the persons that appear in the image. New deep learning proposals are breaking the barriers having very accurate results but with cost of a high processing time. Computer vision tracking algorithms are not accurate as deep learning proposals but reduce the processing time heavily. As final decision, SORT algorithm was selected due to no visual information is used having less processing time and a very simple mathematical solution to track several trackers at the same time with no important increment in processing time.

As the last part reidentification algorithms have been tested in different datasets combination to check the final accuracy and have a robust algorithm in the most possible cases of light or others problem related to image capture. The most complex combination was selected finally to be included in the final image-based tracking framework. Two reidentification steps are performed, the first one is related to the same camera to match lost detections due to occlusions or tracker losses. The second level is the reidentification between camera that don't cross the field of view. Again, experiments show high accuracy in the matching process.

A final step of conversion between pixel coordinate system and real world coordinate system is also presented to have both systems (sensors system and image system) working in the same space obtaining high accuracy in the estimations.

The final part of the complete proposed system combines the information from the sensors

system and the image system to associate the trajectories to the same individuals and fuse the information of the trajectories. The final accuracy is improved thanks to this combination. To end, several real applications have been presented where the system, or parts of the system proposed have been used and tested in similar problems or problems of different nature showing the potential use of the proposed solution.

9.2 Future lines

As future work, a study of the impact of varying the number of nodes in the network can be performed. Moreover, a deep analysis of techniques for Fingerprint auto-recalibration of over time can yield fast and adaptive systems that respond to the indoor scenario changes. Additionally, the research focus on the data fusion of multiple modalities from WSN with visual and depth sensors is relevant to overcome the problems associated to these sensors. An appropriate fusion can solve several tracking problems with cameras such as occlusions, lighting conditions and coverage among others.

More studies regarding generated images such as resolution and complexity of the network can be explored. Further, a bigger area can be also used to assess the methods. Moreover, other Deep Learning techniques such as Reinforcement Learning can be a new possibility to teach the network how to adapt the Fingerprinting database automatically during real time tests. Moreover, the combination of these techniques with other types of sensors like image devices is an interesting field giving more accuracy to the system.

Regarding cameras system, a complete new framework can be developed to asses all the algorithm parts (detection, tracking and reidentification) in only one common framework that can learn directly from the datasets. Regarding tracking problem, it is possible explore more strategies to mitigate detection errors. Although modern detectors are constantly reaching better and better performances, they are still prone to produce a significant number of false negatives and false positives in complex scenarios such as dense pedestrian tracking. Some algorithms have provided solutions to reduce the exclusive reliance on detections by integrating them with information extracted from other sources, but further strategies should be investigated.

Applying deep learning to track different targets is another interesting part (such as in the face tracking application proposed) most of deep learning based tracking algorithms have focused on pedestrian tracking. Since different types of targets pose different challenges, possible

improvements in tracking vehicles, animals, or other objects with the use of deep networks should be investigated. Investigate bounding box regression to solve the problems due to has been shown to be a promising step in obtaining a higher accuracy score, but this has not yet been explored in detail and further improvements should be investigated, for example the use of past and future information to guide the regression.

In deep learning object detection algorithms tasks such as salient object detection, or attention mechanisms can be used to focus better the learning in the main objects under study. Small object detection can be improved for localization accuracy on small objects under partial occlusions, then it is necessary to modify network architectures. The information from different modalities, such as text, thermal data and images, can be fused together to achieve a more discriminant network. To increase the robustness to scale changes, it is demanded to train scale-invariant, multi-scale or scale adaptive detectors. Cascade networks are another possible solution to improve accuracy and are built in different stages or layers to easily distinguish examples that can be rejected at shallow layers so that features and classifiers at latter stages can handle more difficult samples with the aid of the decisions. Finally, unsupervised and weakly supervised learning can be used to automatically label categories.

Investigating in the post-tracking processing, it is possible to apply correction algorithms on the output of a tracker to increase its performance like in the ideas presented in this work. In reidentification algorithms can be improved a deep learning solution to guide the association process.

Finally, the fields of application are very extended and the proposed framework with little modifications can be used to different purposes. Examples of ideas using the complete or parts of the proposed framework can be monitor tourists in a multi-camera environment, track animals in big areas using sensors technologies, mount 360° degree cameras in drones to monitor different parts in a concrete environment, monitor activities in retail sector with both cameras and sensors, sports and lots of other examples.

Bibliography

- [1] M. Eye, “Mynt eye cameras.” <https://www.mynteye.com/pages/mynt-eye-d>. ”[Web; accessed 2019-07-18]”.
- [2] GoPro, “Gopro omni cameras.” https://es.gopro.com/help/articles/question_answer/GoPro-Omni-FAQ. ”[Web; accessed 2019-07-18]”.
- [3] Insta360, “Insta360 cameras.” <https://www.insta360.com/product/insta360-nanos>. ”[Web; accessed 2019-07-18]”.
- [4] 360fly, “360fly cameras.” <https://www.360fly.com/>. ”[Web; accessed 2019-07-18]”.
- [5] Q. Li, H. Qu, Z. Liu, N. Zhou, W. Sun, and J. Li, “Af-dcgan: Amplitude feature deep convolutional gan for fingerprint construction in indoor localization system,” *ArXiv*, vol. abs/1804.05347, 2018.
- [6] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. F. Abdelzaher, “Deepsense: A unified deep learning framework for time-series mobile sensing data processing,” in *WWW*, 2016.
- [7] V. Lehtola, H. Huttunen, F. Christophe, and T. Mikkonen, “Evaluation of visual tracking algorithms for embedded devices,” in *SCIA*, 2017.
- [8] “Ips market expectation by 2025.” <https://www.grandviewresearch.com/industry-analysis/vision-positioning-system-market>. ”[Web; accessed 2019-07-18]”.
- [9] “Ipin market expectation by system.” <https://www.alliedmarketresearch.com/indoor-positioning-and-indoor-navigation-ipin-market>. ”[Web; accessed 2019-07-18]”.

- [10] W. E. Sakpere, M. A. Oshin, and N. Mlitwa, "A state-of-the-art survey of indoor positioning and navigation systems and technologies," 2017.
- [11] S. Ibrahim, "A comprehensive review on intelligent surveillance systems," 2016.
- [12] Microsoft, "Kinect for windows." <https://developer.microsoft.com/es-es/windows/kinect>. "[Web; accessed 2019-07-18]".
- [13] Asus, "Primense xtion pro cameras." <http://xtionprolive.com/primesense-carmin-1.09-3d-sensor-motion-camera>. "[Web; accessed 2019-07-18]".
- [14] Intel, "Intel realsense cameras." <https://software.intel.com/realsense>. "[Web; accessed 2019-07-18]".
- [15] Picoflex, "Picoflex cameras." <https://pmdtec.com/picofamily/>. "[Web; accessed 2019-07-18]".
- [16] D. Macii, F. Trenti, and P. Pivato, "A robust wireless proximity detection technique based on rss and tof measurements," in *2011 IEEE International Workshop on Measurements and Networking Proceedings (M N)*, pp. 31–36, Oct 2011.
- [17] T. J. Pierson, T. Peters, R. A. Peterson, and D. Kotz, "Proximity detection with single-antenna iot devices," 2018.
- [18] M. Lowton, J. M. Brown, J. Finney, and G. Kortuem, "Proximity sensing using ieee 802.15.4 radios," in *EuroSSC*, 2006.
- [19] M. F. i Azam and M. N. Ayyaz, "Location and position estimation in wireless sensor networks," *ArXiv*, vol. abs/1611.03420, 2013.
- [20] R. LeelavathyS and S. Sophia, "Providing localization using triangulation method in wireless sensor networks," 2014.
- [21] B. Guan and X. Li, "An rssi-based wireless sensor network localization algorithm with error checking and correction," *International Journal of Online Engineering (iJOE)*, vol. 13, p. 52, 12 2017.
- [22] G. Ottoy and L. De Strycker, "An improved 2d triangulation algorithm for use with linear arrays," *IEEE Sensors Journal*, vol. 16, pp. 8238–8243, Dec 2016.

- [23] R. P and M. L. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, vol. 1, pp. 374–382, Sep. 2006.
- [24] M. Khalaf-Allah, "Time of arrival (toa)-based direct location method," 06 2015.
- [25] H. Kong, Y. Kwon, and T. Sung, "Comparisons of tdoa triangulation solutions for indoor positioning," 01 2004.
- [26] X. Hui, C. Zhiyuan, Y. Beiya, and N. Rongpei, "Tdoa localization algorithm with compensation of clock offset for wireless sensor networks," *China Communications*, vol. 12, pp. 193–201, 2015.
- [27] S. Chen, J. Zhang, Y. Mao, C. Xu, and Y. Gu, "Efficient distributed method for nlos cooperative localization in wsns," in *Sensors*, 2019.
- [28] J. Kuang, X. Niu, P. Zhang, and X. Chen, "Indoor positioning based on pedestrian dead reckoning and magnetic field matching for smartphones," in *Sensors*, 2018.
- [29] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 1281–1293, Third 2013.
- [30] F. Zampella, A. R. J. Ruiz, and F. S. Granja, "Indoor positioning using efficient map matching, rss measurements, and an improved motion model," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 1304–1317, 2015.
- [31] H. Tran, S. Pandey, and N. Bulusu, "Online map matching for passive indoor positioning systems," in *IWGS@SIGSPATIAL*, 2017.
- [32] S. Mahfouz, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi, "Non-parametric and semi-parametric rssi/distance modeling for target tracking in wireless sensor networks," *IEEE Sensors Journal*, vol. 16, pp. 2115–2126, 2016.
- [33] M. H. Habaebi, R. O. Khamis, A. Zyoud, and M. R. Islam, "Rss based localization techniques for zigbee wireless sensor network," *2014 International Conference on Computer and Communication Engineering*, pp. 72–75, 2014.
- [34] Z. Li, D. B. Acuna, Z. Zhao, J. L. Carrera, and T. Braun, "Fine-grained indoor tracking by fusing inertial sensor and physical layer information in wlans," *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2016.

- [35] H. Yang, R. Zhang, J. Bordoy, F. Hoflinger, W. Li, C. Schindelhauer, and L. M. Reindl, "Smartphone-based indoor localization system using inertial sensor and acoustic transmitter/receiver," *IEEE Sensors Journal*, vol. 16, pp. 8051–8061, 2016.
- [36] S. Cai, W. Liao, C. Luo, M. Li, X. Huang, and P. Li, "Cril: An efficient online adaptive indoor localization system," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 4148–4160, 2017.
- [37] X. He, J. Li, and D. N. Aloï, "Wifi based indoor localization with adaptive motion model using smartphone motion sensors," *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 786–791, 2014.
- [38] D. C. Năstac, F. A. Iftimie, O. Arsene, V. L. M. Ilian, and B. Cramariuc, "Indoor positioning wlan based fingerprinting as supervised machine learning problem," *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 194–199, 2017.
- [39] J. Y. Zhu, A. X. Zheng, J. Xu, and V. O. K. Li, "Spatio-temporal (s-t) similarity model for constructing wifi-based rssi fingerprinting map for indoor localization," *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 678–684, 2014.
- [40] A. Farshad, J. Li, M. K. Marina, and F. J. Garcia, "A microscopic look at wifi fingerprinting for indoor mobile phone localization in diverse environments," *International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–10, 2013.
- [41] C. Luo, H. Hong, M. C. Chan, J. Li, X. Zhang, and Z. Ming, "Mpiloc: Self-calibrating multi-floor indoor localization exploiting participatory sensing," *IEEE Transactions on Mobile Computing*, vol. 17, pp. 141–154, 2018.
- [42] L. Molina, T. Kerdoncuff, D. Shehadeh, N. Montavont, and A. Blanc, "Wmsp: Bringing the wisdom of the crowd to wifi networks," *IEEE Transactions on Mobile Computing*, vol. 16, pp. 3580–3591, 2017.
- [43] S. He, S.-H. G. Chan, L. Yu, and N. Liu, "Maxlifd: Joint maximum likelihood localization fusing fingerprints and mutual distances," *IEEE Transactions on Mobile Computing*, vol. 18, pp. 602–617, 2019.
- [44] H. Ahmadi and R. Bouallegue, "Exploiting machine learning strategies and rssi for localization in wireless sensor networks: A survey," *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1150–1154, 2017.

- [45] J. Torres-Sospedra, R. Montoliu, A. M. Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, “Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems,” *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 261–270, 2014.
- [46] X. Wang, L. Gao, S. Mao, and S. Pandey, “Deepfi: Deep learning for indoor fingerprinting using channel state information,” *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1666–1671, 2015.
- [47] C.-S. Chen, “Artificial neural network for location estimation in wireless communication systems,” in *Sensors*, 2012.
- [48] A. B. Adege, L. Yen, H.-P. Lin, Y. Yayeh, Y. Li, S.-S. Jeng, and G. Berie, “Applying deep neural network (dnn) for large-scale indoor localization using feed-forward neural network (ffnn) algorithm,” *2018 IEEE International Conference on Applied System Invention (ICASI)*, pp. 814–817, 2018.
- [49] M. V. M. Cano, M. A. Zamora-Izquierdo, J. Santa, and A. F. Gómez-Skarmeta, “An indoor localization system based on artificial neural networks and particle filters applied to intelligent buildings,” *Neurocomputing*, vol. 122, pp. 116–125, 2013.
- [50] Y.-K. Cheng, H.-J. Chou, and R. Y. Chang, “Machine-learning indoor localization with access point selection and signal strength reconstruction,” *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2016.
- [51] A. Yassin, Y. Nasser, M. Awad, A. Y. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, “Recent advances in indoor localization: A survey on theoretical approaches and applications,” *IEEE Communications Surveys and Tutorials*, vol. 19, pp. 1327–1346, 2017.
- [52] K. S. Kim, R. Wang, Z. Zhong, Z. Tan, H. Song, J. Cha, and S. Lee, “Large-scale location-aware services in access: Hierarchical building/floor classification and location estimation using wi-fi fingerprinting based on deep neural networks,” 2017.
- [53] M. Nowicki and J. Wietrzykowski, “Low-effort place recognition with wifi fingerprints using deep learning,” 11 2016.
- [54] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. J. Gu, “Deep neural networks for wireless localization in indoor and outdoor environments,” *Neurocomputing*, vol. 194, pp. 279–287, 2016.

- [55] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting," *Big Data Analytics*, vol. 3, pp. 1–17, 2018.
- [56] A. J. C. Moreira, M. J. Nicolau, F. Meneses, and A. Costa, "Wi-fi fingerprinting in the real world - rtls@um at the evaal competition," *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–10, 2015.
- [57] X. Wang, L. Gao, S. Mao, and S. Pandey, "Csi-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 763–776, 2016.
- [58] D. I. Tapia, J. Bajo, J. De Paz, R. Alonso, S. Rodríguez, and J. Corchado Rodríguez, "Using multi-layer perceptrons to enhance the performance of indoor rtls," 01 2011.
- [59] S. Y. M. Vaghefi and R. M. Vaghefi, "A novel multilayer neural network model for toa-based localization in wireless sensor networks," *The 2011 International Joint Conference on Neural Networks*, pp. 3079–3084, 2011.
- [60] M. Dakkak, B. Daachi, A. Nakib, and P. Siarry, "Multi-layer perceptron neural network and nearest neighbor approaches for indoor localization," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1366–1373, 2014.
- [61] J. Y. Zhu, J. Xu, A. X. Zheng, J. He, C. Wu, and V. O. K. Li, "Wifi fingerprinting indoor localization system based on spatio-temporal (s-t) metrics," *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 611–614, 2014.
- [62] Z. E. Khatab, A. Hajihoseini, and S. A. Ghorashi, "A fingerprint method for indoor localization using autoencoder based deep extreme learning machine," *IEEE Sensors Letters*, vol. 2, pp. 1–4, 2018.
- [63] X. Wang, L. Gao, and S. Mao, "Csi phase fingerprinting for indoor localization with a deep learning approach," *IEEE Internet of Things Journal*, vol. 3, pp. 1113–1123, 2016.
- [64] W. Shao, H. Luo, F. Zhao, Y. Ma, Z. Zhao, and A. Crivello, "Indoor positioning based on fingerprint-image and deep learning," *IEEE Access*, vol. 6, pp. 74699–74712, 2018.
- [65] Z. Gu, Z. Chen, Y. Zhang, Y. Zhu, M. Lu, and A. Chen, "Reducing fingerprint collection for indoor localization," *Computer Communications*, vol. 83, pp. 56–63, 2016.

- [66] Y. Shu, Y. Huang, J. Zhang, P. Coue, P. Cheng, J. Chen, and K. G. Shin, "Gradient-based fingerprinting for indoor localization and tracking," *IEEE Transactions on Industrial Electronics*, vol. 63, pp. 2424–2433, 2016.
- [67] J. Jun, L. He, Y. Gu, W. Jiang, G. Kushwaha, A. Vipin, L. Cheng, C. Liu, and T. Zhu, "Low-overhead wifi fingerprinting," *IEEE Transactions on Mobile Computing*, vol. 17, pp. 590–603, 2018.
- [68] A. Belmonte-Hernández, G. Hernández-Peñaloza, F. Álvarez, and G. Conti, "Adaptive fingerprinting in multi-sensor fusion for accurate indoor tracking," *IEEE Sensors Journal*, vol. 17, pp. 4983–4998, 2017.
- [69] S. Yoon, K. Lee, Y. Yun, and I. Rhee, "Acmi: Fm-based indoor localization via autonomous fingerprinting," *IEEE Transactions on Mobile Computing*, vol. 15, pp. 1318–1332, 2016.
- [70] J. Niu, B. Wang, L. Shu, T. Q. Duong, and Y. Chen, "Zil: An energy-efficient indoor localization system using zigbee radio to detect wifi fingerprints," *IEEE Journal on Selected Areas in Communications*, vol. 33, pp. 1431–1442, 2015.
- [71] A. Eleryan, M. Elsabagh, and M. Youssef, "Synthetic generation of radio maps for device-free passive localization," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–5, 2011.
- [72] W. R. Ouyang, A. K.-S. Wong, C.-T. A. Lea, and M. Chiang, "Indoor location estimation with reduced calibration exploiting unlabeled data via hybrid generative/discriminative learning," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 1613–1626, 2012.
- [73] J. Yin, Q. Yang, and L. M. Ni, "Adaptive temporal radio maps for indoor location estimation," *Third IEEE International Conference on Pervasive Computing and Communications*, pp. 85–94, 2005.
- [74] D. Milioris, M. Bradonjic, and P. Mühlethaler, "Building complete training maps for indoor location estimation," *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 75–76, 2015.
- [75] F. Lemic, V. Handziski, G. Caso, L. D. Nardis, and A. Wolisz, "Enriched training database for improving the wifi rssi-based indoor fingerprinting performance," *2016 13th IEEE Annual Consumer Communications and Networking Conference (CCNC)*, pp. 875–881, 2016.

- [76] C. He, S. Guo, Y. Wu, and Y. Yang, “A novel radio map construction method to reduce collection effort for indoor localization,” 2016.
- [77] Y. Cho, J. Kim, M. Ji, Y. Lee, and S. Park, “Gpr based wi-fi radio map construction from real/virtual indoor dynamic surveying data,” *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, pp. 712–714, 2013.
- [78] R. Zhang, A. Bannoura, F. Hoffinger, L. M. Reindl, and C. Schindelhauer, “Indoor localization using a smart phone,” *2013 IEEE Sensors Applications Symposium Proceedings*, pp. 38–42, 2013.
- [79] W. Kang and Y. Han, “Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization,” *IEEE Sensors Journal*, vol. 15, pp. 2906–2916, 2015.
- [80] M. Edel and E. Köppe, “Binarized-blstm-rnn based human activity recognition,” in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, Oct 2016.
- [81] B. Wagstaff, V. Peretroukhin, and J. Kelly, “Improving foot-mounted inertial navigation through real-time motion classification,” *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, 2017.
- [82] Y.-L. Hsu, J.-S. Wang, and C.-W. Chang, “A wearable inertial pedestrian navigation system with quaternion-based extended kalman filter for pedestrian localization,” *IEEE Sensors Journal*, vol. 17, pp. 3193–3206, 2017.
- [83] H. Zhang, W. Yuan, Q. Shen, T. Li, and H. Chang, “A handheld inertial pedestrian navigation system with accurate step modes and device poses recognition,” *Sensors Journal, IEEE*, vol. 15, pp. 1421–1429, 03 2015.
- [84] Z. Zhang, H. Wang, Z. Zhao, and Z. Wu, “A svm-based adaptive stance detection method for pedestrian inertial navigation,” in *2016 10th International Conference on Sensing Technology (ICST)*, pp. 1–6, Nov 2016.
- [85] P. Kasebzadeh, G. Hendeby, C. Fritsche, F. Gunnarsson, and F. Gustafsson, “Imu dataset for motion and device mode classification,” *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, 2017.
- [86] M. Elhoushi, J. Georgy, A. Noureldin, and M. J. Korenberg, “Motion mode recognition for indoor pedestrian navigation using portable devices,” *IEEE Transactions on Instrumentation and Measurement*, vol. 65, pp. 208–221, 2016.

- [87] G. Panahandeh, N. Mohammadiha, A. Leijon, and P. Händel, “Continuous hidden markov model for pedestrian activity classification and gait analysis,” *IEEE Transactions on Instrumentation and Measurement*, vol. 62, pp. 1073–1083, 2013.
- [88] B. Beaufile, F. Chazal, M. Grelet, and B. Michel, “Stride detection for pedestrian trajectory reconstruction: A machine learning approach based on geometric patterns,” *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–6, 2017.
- [89] M. Edel and E. Koppe, “An advanced method for pedestrian dead reckoning using blstm-rnns,” *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–6, 2015.
- [90] B. Shin, C. Kim, J. Kim, S. W. Lee, C. Kee, H. S. Kim, and T. Lee, “Motion recognition-based 3d pedestrian navigation system using smartphone,” *IEEE Sensors Journal*, vol. 16, pp. 6977–6989, 2016.
- [91] F. Gu, K. Khoshelham, S. Valaee, J. Shang, and R. Q. Zhang, “Locomotion activity recognition using stacked denoising autoencoders,” *IEEE Internet of Things Journal*, vol. 5, pp. 2085–2093, 2018.
- [92] L. L. Shen and W. W. S. Hui, “Improved pedestrian dead-reckoning-based indoor positioning by rssi-based heading correction,” *IEEE Sensors Journal*, vol. 16, pp. 7762–7773, 2016.
- [93] K. Kunze, P. Lukowicz, K. Partridge, and J. Begole, “Which way am i facing: Inferring horizontal device orientation from an accelerometer signal,” *2009 International Symposium on Wearable Computers*, pp. 149–150, 2009.
- [94] A. Belmonte-Hernández, G. Hernández-Peñaloza, D. M. Gutiérrez, and F. Álvarez, “Swiblax: Multi-sensor deep learning fingerprint for precise real-time indoor tracking,” *IEEE Sensors Journal*, vol. 19, pp. 3473–3486, 2019.
- [95] “Ieee 802.11 wireless local area networks.” <http://www.ieee802.org/11/>. ”[Web; accessed 2019-08-10]”.
- [96] VVAA, “Bluetooth protocol.” www.bluetooth.com/develop-with-bluetooth. ”[Web; accessed 2019-08-10]”.
- [97] “Ieee802.15.4 wpan task group 4.” <http://www.ieee802.org/15/pub/TG4.html>. ”[Web; accessed 2019-08-10]”.

- [98] S. Kokalj-Filipovic, L. Greenstein, B. Cheng, and M. Gruteser, "Methods for extracting v2v propagation models from imperfect rssi field data," *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, pp. 1–5, 2015.
- [99] M. Klemm and G. Tröster, "Em energy absorption in the human body tissues due to uwb antennas," 2006.
- [100] M. Singh and P. M. Khilar, "Actuating sensor for determining the direction of arrival using maximal rssi," 2014.
- [101] T. Alhmiedat, G. Samara, and A. O. A. Salem, "An indoor fingerprinting localization approach for zigbee wireless sensor networks," *ArXiv*, vol. abs/1308.1809, 2013.
- [102] B. Shin, J. H. Lee, T. Lee, and H. S. Kim, "Enhanced weighted k-nearest neighbor algorithm for indoor wi-fi positioning systems," *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, vol. 2, pp. 574–577, 2012.
- [103] J. Kim and C. D. Scott, "Robust kernel density estimation," *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3381–3384, 2008.
- [104] V. Akbarzadeh, C. Gagné, and M. Parizeau, "Kernel density estimation for target trajectory prediction," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3449–3456, 2015.
- [105] A. Kushki, K. K. Plataniotis, A. N. Venetsanopoulos, and C. Regazzoni, "Radio map fusion for indoor positioning in wireless local area networks," *2005 7th International Conference on Information Fusion*, vol. 2, pp. 8 pp.–, 2005.
- [106] W. R. Schucany, "Kernel smoothers: An overview of curve estimators for the first graduate course in nonparametric statistics," 2004.
- [107] J. Yim, S. Jeong, J. Joo, and C. Park, "Utilizing map information for wlan-based kalman filter indoor tracking," *2008 Second International Conference on Future Generation Communication and Networking Symposia*, vol. 5, pp. 58–62, 2008.
- [108] S. Feng and R. Murray-Smith, "Fusing kinect sensor and inertial sensors with multi-rate kalman filter," 2014.
- [109] K. Lu, K. C. Chang, and R. Zhou, "The exact algorithm for multi-sensor asynchronous track-to-track fusion," *2015 18th International Conference on Information Fusion (Fusion)*, pp. 886–892, 2015.

- [110] V. Z. Fathabadi, M. Shahbazian, K. Salahshour, and L. Jargani, "Comparison of adaptive kalman filter methods in state estimation of a nonlinear system using asynchronous measurements," 2009.
- [111] P. Guo, X. Wang, and Y. Wu, "Data fusion using weighted likelihood," 2012.
- [112] E. Mirzazadeh and B. Moshiri, "Multi-sensor estimation using ckf and dkf," *ICCCKE 2013*, pp. 321–325, 2013.
- [113] S. A. Hoseini and M. R. Ashraf, "Computational complexity comparison of multi-sensor single target data fusion methods by matlab," *ArXiv*, vol. abs/1307.3005, 2013.
- [114] D. Tenne and T. Singh, "Optimal design of α - β - γ filters," *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 6, pp. 4348–4352 vol.6, 2000.
- [115] Y.-S. Chiou, C.-L. Wang, and S.-C. Yeh, "Reduced-complexity scheme using alpha-beta filtering for location tracking," *IET Communications*, vol. 5, pp. 1806–1813, 2011.
- [116] C. Liu, A. Kiring, N. Salman, L. S. Mihaylova, and I. Esnaola, "A kriging algorithm for location fingerprinting based on received signal strength," *2015 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pp. 1–6, 2015.
- [117] S. O. H. Madgwick, "An efficient orientation filter for inertial and inertial / magnetic sensor arrays," 2010.
- [118] F. S. Panchal and M. Panchal, "Review on methods of selecting number of hidden nodes in artificial neural network," 2014.
- [119] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.
- [120] "Adaptiv: An adaptive jerk pace buffer step detection algorithm." <https://github.com/danielmurray/adaptiv>. "[Web; accessed 2019-08-26]".
- [121] H. Weinberg, "Using the adxl 202 in pedometer and personal navigation applications," 2002.
- [122] Q. Tian, Z. A. Salcic, K. I.-K. Wang, and Y. Pan, "A multi-mode dead reckoning system for pedestrian tracking using smartphones," *IEEE Sensors Journal*, vol. 16, pp. 2079–2093, 2016.

- [123] I. Bylemans, M. Weyn, and M. Klepal, "Mobile phone-based displacement estimation for opportunistic localisation systems," *2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pp. 113–118, 2009.
- [124] M. S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Trans. Signal Processing*, vol. 50, pp. 174–188, 2002.
- [125] T. Li, M. Bolic, and P. M. Djurić, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal Processing Magazine*, vol. 32, pp. 70–86, 2015.
- [126] B. G. Sileshi, C. Ferrer, and J. Oliver, "Particle filters and resampling techniques: Importance in computational complexity analysis," *2013 Conference on Design and Architectures for Signal and Image Processing*, pp. 319–325, 2013.
- [127] L. Martino, V. Elvira, and F. Louzada, "Alternative effective sample size measures for importance sampling," *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 1–5, 2016.
- [128] A. Belmonte-Hernández, G. Hernández-Peñaloza, D. M. Gutiérrez, and F. Álvarez, "Recurrent model for wireless indoor tracking and positioning recovering using generative networks," *IEEE Sensors Journal*, pp. 1–1, 2019.
- [129] T. Bouwmans and B. Garcia-Garcia, "Background subtraction in real applications: Challenges, current models and future directions," *ArXiv*, vol. abs/1901.03577, 2019.
- [130] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, 05 2014.
- [131] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.
- [132] P. Dollár, S. J. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, 2010.
- [133] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1030–1037, 2010.

- [134] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3626–3633, 2012.
- [135] L. Spinello and K. O. Arras, "People detection in rgb-d data," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3838–3843, 2011.
- [136] X. Li, K. Wang, W. Wang, and Y. Li, "A multiple object tracking method using kalman filter," *The 2010 IEEE International Conference on Information and Automation*, pp. 1862–1866, 2010.
- [137] J. Wang, Y. Ma, C. Li, H. Wang, and J. Liu, "An efficient multi-object tracking method using multiple particle filters," *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 6, pp. 568–572, 2009.
- [138] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, p. 13, 2006.
- [139] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36 7, pp. 1442–68, 2014.
- [140] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1834–1848, 2015.
- [141] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC*, 2006.
- [142] B. Babenko, M.-H. Yang, and S. J. Belongie, "Visual tracking with online multiple instance learning," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990, 2009.
- [143] J. F. Henriques, R. Caseiro, P. Martins, and J. P. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *ECCV*, 2012.
- [144] A. Lukezic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *CVPR*, 2017.
- [145] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," *2010 20th International Conference on Pattern Recognition*, pp. 2756–2759, 2010.

- [146] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, 12 2011.
- [147] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, 2010.
- [148] D. Mitzel and B. Leibe, "Real-time multi-person tracking with detector assisted structure propagation," *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 974–981, 2011.
- [149] A. Javed, A. Ejaz, S. Liaqat, A. Ashraf, and M. B. Ihsan, "Automatic target classifier for a ground surveillance radar using linear discriminant analysis and logistic regression," *2012 9th European Radar Conference*, pp. 302–305, 2012.
- [150] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3d range data," in *AAAI*, 2010.
- [151] L. Spinello, M. Luber, and K. O. Arras, "Tracking people in 3d using a bottom-up top-down detector," *2011 IEEE International Conference on Robotics and Automation*, pp. 1304–1310, 2011.
- [152] C. Benedek, "3d people surveillance on range data sequences of a rotating lidar," *Pattern Recognition Letters*, vol. 50, pp. 149–158, 2014.
- [153] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pp. 9–14, 2010.
- [154] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, pp. 611–622, 2011.
- [155] W. Lin, M.-T. Sun, R. Poovendran, and Z. Zhang, "Group event detection with a varying number of group members for video surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, pp. 1057–1067, 2010.
- [156] B. Solmaz, B. E. Moore, and M. Shah, "Identifying behaviors in crowd scenes using stability analysis for dynamical systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2064–2070, 2012.

- [157] H. Idrees, N. Warner, and M. Shah, "Tracking in dense crowds using prominence and neighborhood motion concurrence," *Image Vision Comput.*, vol. 32, pp. 14–26, 2014.
- [158] T. Ko, "A survey on behavior analysis in video surveillance for homeland security applications," *2008 37th IEEE Applied Imagery Pattern Recognition Workshop*, pp. 1–8, 2008.
- [159] D. Gowsikhaa and S. Abirami, "Automated human behavior analysis from surveillance videos: a survey," *Artificial Intelligence Review*, vol. 42, pp. 747–765, 2012.
- [160] N. Kiryati, T. Riklin-Raviv, Y. Ivanchenko, and S. Rochel, "Real-time abnormal motion detection in surveillance video," *2008 19th International Conference on Pattern Recognition*, pp. 1–4, 2008.
- [161] H. Foroughi, H. L. Yazdi, H. R. Pourreza, and M. H. Javidi, "An eigenspace-based approach for human fall detection using integrated time motion image and multi-class support vector machine," *2008 4th International Conference on Intelligent Computer Communication and Processing*, pp. 83–90, 2008.
- [162] S. Park and J. K. Aggarwal, "Semantic-level understanding of human actions and interactions using event hierarchy," *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 12–12, 2004.
- [163] F. Brémond, M. Thonnat, and M. Zúñiga, "Video-understanding framework for automatic behavior recognition," *Behavior Research Methods*, vol. 38, pp. 416–426, 2006.
- [164] Z.-Q. Zhao, P. Zheng, S. tao Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, 2018.
- [165] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2013.
- [166] R. B. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.
- [167] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.
- [168] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

- [169] S. Liu, D. Huang, and Y. Wang, “Receptive field block net for accurate and fast object detection,” *ArXiv*, vol. abs/1711.07767, 2017.
- [170] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015.
- [171] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2016.
- [172] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *ArXiv*, vol. abs/1804.02767, 2018.
- [173] J. Dai, Y. C. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *NIPS*, 2016.
- [174] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2016.
- [175] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, “Local binary convolutional neural networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4284–4293, 2016.
- [176] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask r-cnn,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.
- [177] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.
- [178] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1302–1310, 2016.
- [179] R. A. Güler, N. Neverova, and I. Kokkinos, “Densepose: Dense human pose estimation in the wild,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7297–7306, 2018.
- [180] H. Fang, S. Xie, and C. Lu, “Rmpe: Regional multi-person pose estimation,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2353–2362, 2016.

- [181] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *ECCV*, 2014.
- [182] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2d human pose estimation: New benchmark and state of the art analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [183] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, “Pose Flow: Efficient online pose tracking,” *ArXiv e-prints*, 2018.
- [184] D. Gordon, A. Farhadi, and D. Fox, “Re3: Real-time recurrent regression networks for visual tracking of generic objects,” *IEEE Robotics and Automation Letters*, vol. 3, pp. 788–795, 2017.
- [185] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in *ECCV*, 2016.
- [186] R. O. C. García and O. Aycard, “Multiple sensor fusion and classification for moving object detection and tracking,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 525–534, 2016.
- [187] K.-E. Kim, C.-J. Lee, D.-S. Pae, and M.-T. Lim, “Sensor fusion for vehicle tracking with camera and radar sensor,” *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1075–1077, 2017.
- [188] A. Klausner, A. Teng, and B. Rinner, “Vehicle classification on multi-sensor smart cameras using feature- and decision-fusion,” *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 67–74, 2007.
- [189] C. Chen, R. Jafari, and N. Kehtarnavaz, “Improving human action recognition using fusion of depth camera and inertial sensors,” *IEEE Transactions on Human-Machine Systems*, vol. 45, pp. 51–61, 2015.
- [190] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, “Spatially supervised recurrent convolutional neural networks for visual object tracking,” *arXiv preprint arXiv:1607.05781*, 2016.
- [191] A. Broad, M. P. Jones, and T.-Y. Lee, “Recurrent multi-frame single shot detector for video object detection,” in *BMVC*, 2018.

- [192] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Detect to track and track to detect,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3057–3065, 2017.
- [193] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran, “Detect-and-track: Efficient pose estimation in videos,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 350–359, 2017.
- [194] W. Yang, Y. Qian, J.-K. Kämäräinen, F. Cricri, and L. Fan, “Object detection in equirectangular panorama,” *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2190–2195, 2018.
- [195] F. Deng, X. Zhu, and J. Ren, “Object detection on panoramic images based on deep learning,” *2017 3rd International Conference on Control, Automation and Robotics (IC-CAR)*, pp. 375–380, 2017.
- [196] U. Kart, J.-K. Kämäräinen, L. Fan, and M. Gabbouj, “Evaluation of visual object trackers on equirectangular panorama,” in *VISIGRAPP*, 2018.
- [197] L. Liu, J. Ding, and Q. Liu, “A new panoramic visual tracking method,” in *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 315–319, Oct 2018.
- [198] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, “Omnidepth: Dense depth estimation for indoors spherical panoramas,” in *ECCV*, 2018.
- [199] J. Gutiérrez-Cillán, E. J. David, A. Coutrot, M. P. D. Silva, and P. L. Callet, “Introducing un salient360! benchmark: A platform for evaluating visual attention models for 360° contents,” *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–3, 2018.
- [200] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, “Salnet360: Saliency maps for omnidirectional images with cnn,” *Sig. Proc.: Image Comm.*, vol. 69, pp. 26–34, 2017.
- [201] T. Cohen, M. Geiger, J. Köhler, and M. Welling, “Convolutional networks for spherical signals,” *ArXiv*, vol. abs/1709.04893, 2017.
- [202] Y.-C. Su and K. Grauman, “Flat2sphere: Learning spherical convolution for fast features from 360° imagery,” *ArXiv*, vol. abs/1708.00919, 2017.
- [203] T. Cohen, M. Geiger, J. Köhler, and M. Welling, “Spherical cnns,” *ArXiv*, vol. abs/1801.10130, 2018.

- [204] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, “Learning $so(3)$ equivariant representations with spherical cnns,” in *ECCV*, 2017.
- [205] Y.-C. Su and K. Grauman, “Kernel transformer networks for compact spherical convolution,” *ArXiv*, vol. abs/1812.03115, 2018.
- [206] B. Coors, A. Condurache, and A. Geiger, “Spherenet: Learning spherical representations for detection and classification in omnidirectional images,” in *ECCV*, 2018.
- [207] Y.-C. Su, D. Jayaraman, and K. Grauman, “Pano2vid: Automatic cinematography for watching 360 videos,” 2016.
- [208] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun, “Deep 360 pilot: Learning a deep agent for piloting through 360° sports videos,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1396–1405, 2017.
- [209] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, “Fixation prediction for 360° video streaming in head-mounted virtual reality,” in *NOSSDAV*, 2017.
- [210] A. Nguyen, Z. Yan, and K. Nahrstedt, “Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction,” in *ACM Multimedia*, 2018.
- [211] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang, “Semantic-driven generation of hyperlapse from 360° video,” *ArXiv*, vol. abs/1703.10798, 2017.
- [212] M. Assens, X. Giró, K. McGuinness, and N. E. O’Connor, “Saltinet: Scan-path prediction on 360 degree images using saliency volumes,” *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2331–2338, 2017.
- [213] W. Zhang, R. Zhao, X. Tong, and X. Wang, “Deepreid: Deep filter pairing neural network for person re-identification,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 152–159, 2014.
- [214] E. Ahmed, M. J. Jones, and T. K. Marks, “An improved deep learning architecture for person re-identification,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3908–3916, 2015.
- [215] R. R. Varior, M. Haloi, and G. Wang, “Gated siamese convolutional neural network architecture for human re-identification,” *ArXiv*, vol. abs/1607.08378, 2016.

- [216] Y. Shen, X. Tong, H. Li, S. Yi, and X. Wang, “End-to-end deep kronecker-product matching for person re-identification,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6886–6895, 2018.
- [217] Y. Guo and N.-M. Cheung, “Efficient and deep person re-identification using multi-level similarity,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2335–2344, 2018.
- [218] A. Subramaniam, M. Chatterjee, and A. Mittal, “Deep neural networks with inexact matching for person re-identification,” in *NIPS*, 2016.
- [219] Y. Wang, Z. Chen, F. Wu, and G. Wang, “Person re-identification with cascaded pairwise convolutions,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1470–1478, 2018.
- [220] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, “Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline),” in *ECCV*, 2017.
- [221] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, “Alignedreid: Surpassing human-level performance in person re-identification,” *ArXiv*, vol. abs/1711.08184, 2017.
- [222] Y. Fu, Y. Wei, Y. Zhou, H. Shi, G. Huang, X. Wang, Z. Yao, and T. S. Huang, “Horizontal pyramid matching for person re-identification,” in *AAAI*, 2018.
- [223] G. Wang, Y. Yuan, X. Chen, J. Li, and X. Zhou, “Learning discriminative features with multiple granularities for person re-identification,” in *ACM Multimedia*, 2018.
- [224] W. Li, X. Zhu, and S. Gong, “Harmonious attention network for person re-identification,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2285–2294, 2018.
- [225] J. Si, H. Zhang, C.-G. Li, J. Kuen, X. Kong, A. C. Kot, and G. Wang, “Dual attention matching network for context-aware feature sequence based person re-identification,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5363–5372, 2018.
- [226] C. Song, Y. Huang, W. Ouyang, and L. Wang, “Mask-guided contrastive attention model for person re-identification,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1179–1188, 2018.

- [227] J. Xu, R. Zhao, F. Zhu, H. Wang, and W. Ouyang, "Attention-aware compositional network for person re-identification," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2119–2128, 2018.
- [228] H. Zhao, M. Tian, S. Sun, J. Shao, J. Yan, S. Yi, X. Wang, and X. Tang, "Spindle net: Person re-identification with human body region guided feature decomposition and fusion," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 907–915, 2017.
- [229] C. Su, J. Li, S. Zhang, J. Xing, W. Gao, and Q. Tian, "Pose-driven deep convolutional model for person re-identification," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3980–3989, 2017.
- [230] Y. Suh, J. Wang, S. Tang, T. Mei, and K. M. Lee, "Part-aligned bilinear representations for person re-identification," *ArXiv*, vol. abs/1804.07094, 2018.
- [231] M. Tian, S. Yi, H. Li, S. Li, X. Zhang, J. Shi, J. Yan, and X. Wang, "Eliminating background-bias for robust person re-identification," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5794–5803, 2018.
- [232] W. Li, X. Zhu, and S. Gong, "Person re-identification by deep joint learning of multi-loss classification," in *IJCAI*, 2017.
- [233] D. Li, X. Chen, Z. Zhang, and K. Huang, "Learning deep context-aware features over body and latent parts for person re-identification," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7398–7407, 2017.
- [234] L. Zhao, X. Li, Y. Zhuang, and J. Wang, "Deeply-learned part-aligned representations for person re-identification," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3239–3248, 2017.
- [235] X. Chang, T. M. Hospedales, and T. Xiang, "Multi-level factorisation net for person re-identification," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2109–2118, 2018.
- [236] X. Liu, H. Zhao, M. Tian, L. Sheng, J. Shao, S. Yi, J. Yan, and X. Wang, "Hydraplus-net: Attentive deep features for pedestrian analysis," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 350–359, 2017.
- [237] Q. Yu, X. Chang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "The devil is in the middle: Exploiting mid-level representations for cross-domain instance matching," *ArXiv*, vol. abs/1711.08106, 2017.

- [238] Y. Wang, L. Wang, Y. You, X. Zou, V. Chen, S. Li, G. Huang, B. Hariharan, and K. Q. Weinberger, "Resource aware person re-identification across multiple resolutions," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8042–8051, 2018.
- [239] S. Wu, Y.-C. Chen, X. Li, A. Wu, J. You, and W.-S. Zheng, "An enhanced deep feature representation for person re-identification," *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8, 2016.
- [240] B. Ma, Y. Su, and F. Jurie, "Local descriptors encoded by fisher vectors for person re-identification," in *ECCV Workshops*, 2012.
- [241] L. Wu, C. Shen, and A. van den Hengel, "Deep linear discriminant analysis on fisher networks: A hybrid architecture for person re-identification," *Pattern Recognition*, vol. 65, pp. 238–250, 2016.
- [242] T. Xiao, H. Li, W. Ouyang, and X. Wang, "Learning deep feature representations with domain guided dropout for person re-identification," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1249–1258, 2016.
- [243] Y. Dong, Z. Lei, and S. Z. Li, "Deep metric learning for practical person re-identification," *ArXiv*, vol. abs/1407.4979, 2014.
- [244] J. Wang, Z. Wang, C. Gao, N. Sang, and R. Huang, "Deeplist: Learning deep features with adaptive listwise constraint for person reidentification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, pp. 513–524, 2017.
- [245] F. Wang, W. Zuo, L. Lin, D. Zhang, and L. Zhang, "Joint learning of single-image and cross-image representations for person re-identification," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1288–1296, 2016.
- [246] H. Liu, J. Feng, M. Qi, J. Jiang, and S. Yan, "End-to-end comparative attention networks for person re-identification," *IEEE Transactions on Image Processing*, vol. 26, pp. 3492–3506, 2017.
- [247] S.-Z. Chen, C.-C. Guo, and J. Lai, "Deep ranking for person re-identification via joint representation learning," *IEEE Transactions on Image Processing*, vol. 25, pp. 2353–2367, 2016.
- [248] A. Franco and L. Oliveira, "A coarse-to-fine deep learning for person re-identification," *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–7, 2016.

- [249] A. Franco and L. Oliveira, “Convolutional covariance features: Conception, integration and performance in person re-identification,” *Pattern Recognition*, vol. 61, pp. 593–609, 2017.
- [250] S. Wang, C. Zhang, L. Duan, L. Wang, S. Wu, and L. Chen, “Person re-identification based on deep spatio-temporal features and transfer learning,” *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1660–1665, 2016.
- [251] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W.-S. Zheng, and S. Z. Li, “Embedding deep metric for person re-identification a study against large variations,” in *ECCV*, 2016.
- [252] N. McLaughlin, J. M. del Rincón, and P. C. Miller, “Person reidentification using deep convnets with multitask learning,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, pp. 525–539, 2017.
- [253] S. Iodice, A. Petrosino, and I. Ullah, “Strict pyramidal deep architectures for person re-identification,” 2015.
- [254] D. Tao, Y. Guo, B. Yu, J. Pang, and Z. Yu, “Deep multi-view feature learning for person re-identification,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, pp. 2657–2666, 2018.
- [255] C. Mao, Y. Li, Z. Zhang, Y. Zhang, and X. Li, “Pyramid person matching network for person re-identification,” in *ACML*, 2017.
- [256] X. Qian, Y. Fu, Y.-G. Jiang, T. Xiang, and X. Xue, “Multi-scale deep learning architectures for person re-identification,” *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5409–5418, 2017.
- [257] “Voc-pascal dataset.” <http://host.robots.ox.ac.uk/pascal/VOC/>. ”[Web; accessed 2019-09-02]”.
- [258] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.
- [259] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.
- [260] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 1mb model size,” *ArXiv*, vol. abs/1602.07360, 2017.

- [261] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2014.
- [262] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.
- [263] "Vivotek fd8379-hv camera." <https://www.vivotek.com/fd8379-hv>. "[Web; accessed 2019-09-09]".
- [264] "Vivotek fe8180 camera." <https://www.vivotek.com/fe8180>. "[Web; accessed 2019-09-09]".
- [265] "Ricoh theta v camera." <https://theta360.com/es/about/theta/v.html>. "[Web; accessed 2019-09-09]".
- [266] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 1330–1334, 2000.
- [267] J.Y.Bouguet, "Matlab calibration tool." http://www.vision.caltech.edu/bouguetj/calib_doc. "[Web; accessed 2019-10-18]".
- [268] H. Cantzler, "Random sample consensus (ransac)," 1981.
- [269] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1116–1124, 2015.
- [270] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.
- [271] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer gan to bridge domain gap for person re-identification," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 79–88, 2017.
- [272] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensemble of localized features," in *ECCV*, 2008.
- [273] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof, "Person re-identification by descriptive and discriminative classification," in *SCIA*, 2011.

- [274] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *ECCV*, 2016.
- [275] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 583–596, 2015.
- [276] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft, "Simple online and realtime tracking," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.
- [277] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," *ArXiv*, vol. abs/1606.09549, 2016.
- [278] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, 2017.
- [279] Z. Cao, G. H. Martinez, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, 2018.
- [280] D. Zhang, G. Guo, D. Huang, and J. Han, "Poseflow: A deep motion representation for understanding human behaviors in videos," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6762–6770, 2018.
- [281] M. Andriluka, L. Pishchulin, P. V. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3686–3693, 2014.
- [282] Y. Liu, Y. Wang, S. Wang, T. Liang, Q. Zhao, Z. Tang, and H. Ling, "Cb-net: A novel composite backbone network architecture for object detection," *ArXiv*, vol. abs/1909.03625, 2019.
- [283] G. Hernández-Peñaloza, A. Belmonte-Hernández, M. Quintana, and F. Álvarez, "A multi-sensor fusion scheme to increase life autonomy of elderly people with cognitive problems," *IEEE Access*, vol. 6, pp. 12775–12789, 2018.
- [284] T. D. Gottschalk, "Concurrent implementation of munkres algorithm," *Proceedings of the Fifth Distributed Memory Computing Conference, 1990.*, pp. 52–57, 1990.

- [285] A. Belmonte-Hernández, V. Solachidis, T. Theodoridis, G. Hernández-Peñaloza, G. Conti, N. Vretos, F. Álvarez, and P. Daras, "Person tracking association using multi-modal systems," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Aug 2017.
- [286] A. Alaiad and L. Zhou, "Patients' adoption of wsn-based smart home healthcare systems: An integrated model of facilitators and barriers," *IEEE Transactions on Professional Communication*, vol. 60, pp. 4–23, 2017.
- [287] F. A. Moreno, J. A. Merchán-Baeza, M. González-Sánchez, J. González, and A. I. Cuesta-Vargas, "Experimental validation of depth cameras for the parameterization of functional balance of patients in clinical tests," in *Sensors*, 2017.
- [288] R. Lun, C. Gordon, and W. Zhao, "Tracking the activities of daily lives: An integrated approach," *2016 Future Technologies Conference (FTC)*, pp. 466–475, 2016.
- [289] F. Alvarez, M. Popa, N. Vretos, A. Belmonte-Hernández, S. Asteriadis, V. Solachidis, T. Mariscal, D. Dotti, and P. Daras, "Multimodal monitoring of parkinson's and alzheimer's patients using the ict4life platform," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Aug 2017.
- [290] F. Alvarez, M. Popa, V. Solachidis, G. Hernández-Peñaloza, A. Belmonte-Hernández, S. Asteriadis, N. Vretos, M. Quintana, T. Theodoridis, D. Dotti, and P. Daras, "Behavior analysis through multimodal sensing for care of parkinson's and alzheimer's patients," *IEEE MultiMedia*, vol. 25, pp. 14–25, Jan 2018.
- [291] R. Malhotra, "Hybrid broadcast broadband tv: The way forward for connected tvs," *IEEE Consumer Electronics Magazine*, vol. 2, pp. 10–16, 2013.
- [292] V. Jain and E. G. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," 2010.
- [293] S. Uribe, A. Belmonte, F. Moreno, A. Llorente, J. P. Lopez, and F. Alvarez, "New access services in hbbtv based on a deep learning approach for media content analysis," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, p. 1–17, 2019.