Multi-Sensor Fusion for Autonomous Resilient Perception Exploiting Classical and Deep Learning Techniques

Thesis · April 2024

DOI: 10.13140/RG.2.2.29422.42560

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

PhD in Computer Science, Control and Geoinformation

XXXVI PhD Cycle

Multi-Sensor Fusion for Autonomous Resilient Perception

Exploiting Classical and Deep Learning Techniques

Fabrizio Romanelli

A.Y. 2023/2024

Tutor: Prof. Francesco Martinelli

Coordinator: Prof. Francesco Quaglia

Nascosta in qualche angolo di questo mare
c'è la nostra isola più bella
e mai trovata.
A papà.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Fabrizio Romanelli
A.Y. 2023/2024
March 2024

Acknowledgements

I want to say thank to the inspiration that comes from incidents and melancholy.
Thanks to all the souls who crossed my life, both those who decided to live a sheltered, ordinary existence and those who have chosen the path of incertitude, imperfection and jeopardy.
I am still wandering that path, you are not alone.

Abstract

This Ph.D. thesis entitled *Multi-Sensor Fusion for Autonomous Resilient Perception Exploiting Classical and Deep Learning Techniques* addresses the critical challenges associated with perception systems in autonomous applications. Perception plays a vital role in enabling autonomous systems to understand and interpret the environment, making accurate decisions and ensuring the safety and reliability of these systems. However, perception is inherently complex due to various sources of uncertainty, including sensor noise, occlusions, varying lighting conditions, and dynamic environments.

To overcome these challenges, this research focuses on the development of robust and resilient perception systems through the fusion of data from multiple sensors. The fusion of information from diverse sensors can provide complementary and redundant information, enhancing the overall perception performance and increasing resilience to sensor failures or limitations. The thesis investigates both classical and deep learning techniques for sensor fusion, leveraging their respective strengths to improve perception accuracy and reliability.

The classical techniques explored in this research include probabilistic methods, such as Bayesian filtering and Kalman filtering, which enable the integration of sensor measurements and estimation of the state of the environment. These techniques are enhanced with advanced methodologies to overcome the problems related to sensor degradation and sensor unavailability (e.g. due to breakdowns) and to handle complex real-life scenarios with multiple moving objects and occlusions. Additionally, optimization algorithms are employed to further improve the performance of the sensor fusion process.

Moreover, deep learning techniques, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are investigated for their capability to learn complex representations and patterns from sensor data. Deep learning models are trained on large-scale datasets to recognize and classify objects, detect anomalies, and estimate the environment state. The fusion of deep learning-based outputs with classical techniques allows for a more comprehensive and accurate understanding of the environment.

Furthermore, the thesis addresses the issue of resilience in perception systems by incorporating fault detection and recovery mechanisms. Robustness against sensor failures, sensor drift, and adversarial attacks is achieved through the integration of redundancy and

outlier rejection techniques. The proposed methods enable the perception system to adapt to changing conditions and maintain reliable performance, even in the presence of sensor abnormalities (e.g. malfunctioning). To be more specific, this Ph.D. thesis focuses on a particular perception problem, namely SLAM (Simultaneous Localization And Mapping), applied to a variety of contexts such as mobile robots and generic agents moving in unknown environments while acquiring measurements coming from different set of sensors (odometry, Ultra Wide Band, Ultra High Frequency - RFID, visual systems). The applications presented in this thesis are related to the following specific cases:

- Odometry-UHF RFID system
- Visual-UWB system
- Visual-UWB system with deep learning approach
- Multiple visual system.

The effectiveness of the proposed multi-sensor fusion approaches is evaluated through extensive experiments and simulations using real-world datasets and synthetic scenarios. The evaluation encompasses various autonomous applications, including autonomous driving, robotics, and surveillance systems. The results demonstrate significant improvements in perception accuracy, robustness, and resilience compared to single-sensor or naive fusion approaches.

In conclusion, this Ph.D. thesis contributes to the field of autonomous perception by presenting novel multi-sensor fusion techniques that exploit both classical and deep learning approaches. The research advances the state-of-the-art in perception systems by improving accuracy, resilience, and adaptability, thus paving the way for more reliable and trustworthy autonomous systems in diverse real-world applications.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Problem statement: need for resilient sensor fusion

The field of autonomous systems has witnessed tremendous growth in recent years, thanks to advancements in sensor technology, machine learning, and artificial intelligence. Autonomous systems are used in various applications such as robotics, self-driving cars, unmanned aerial vehicles, and many others. The primary objective of autonomous systems is to perform tasks without human intervention or control.

One of the key elements that enable autonomous systems to operate effectively is sensor fusion. Sensor fusion involves combining data from different sensors to obtain a more accurate and comprehensive representation of the environment. In other words, sensor fusion aims to provide a complete and reliable picture of the surrounding environment to enable autonomous systems to make informed decisions.

However, relying solely on sensor data can be challenging, as sensors can fail or provide erroneous data. This can lead to incorrect decisions and potentially hazardous situations. Therefore, there is a need for resilient sensor fusion to ensure that autonomous systems can operate effectively in dynamic and unpredictable environments.

The problem statement of this thesis is the need for resilient sensor fusion for autonomous systems. In this thesis, we will explore the challenges associated with sensor fusion in autonomous systems, including sensor failures and data uncertainties. We will also investigate the state-of-the-art approaches for resilient sensor fusion and propose new techniques to improve the robustness and reliability of autonomous systems.

The rest of the thesis is organized as follows. Chapter 2 provides an overview of the resilient robot perception topic and of the existing literature on resilient sensor fusion and the challenges associated with it. Chapter 3 presents the proposed methodology for resilient sensor fusion. Chapter 4 shows the framework for resilient multi-sensor fusion. Chapter 5

presents the experimental results and analysis of the proposed approach. Chapter 6 shows the impact of research, analyzing the resulting publications. Furthermore, Chapter 7 reports a discussion about the implications of the presented research on the society. Finally, Chapter 8 concludes the thesis and outlines the future directions for research in the field of resilient sensor fusion for autonomous systems.

1.2 Context

The emergence of autonomous systems has been one of the most significant technological advancements of the 21st century. Autonomous systems are becoming increasingly prevalent in many applications, including transportation, agriculture, healthcare, and security. These systems are designed to operate independently, using a combination of sensors, algorithms, and decision-making systems. However, the reliability of these systems is heavily dependent on the accuracy and consistency of the sensor data they receive. Sensor data can be affected by many factors such as environmental conditions, sensor malfunctions, data uncertainties, and cyber-attacks.

Sensor fusion is a crucial component of autonomous systems that aims to integrate data from different sensors to provide a more accurate and reliable representation of the environment. This integration can be performed at various levels, such as raw sensor data fusion, feature-level fusion, and decision-level fusion. The primary goal of sensor fusion is to improve the accuracy and reliability of autonomous systems by reducing the impact of sensor failures, data uncertainties, and other potential sources of error.

However, achieving resilient sensor fusion is a challenging task, as there are several factors that can affect the reliability of the system. One of the primary challenges is sensor failures, which can occur due to hardware malfunction, power loss, or other factors. Sensor failures can result in incomplete or incorrect data, which can lead to incorrect decisions and potential hazards. Another challenge is data uncertainties, which can arise due to noise, interference, or environmental factors. These uncertainties can cause the system to make incorrect decisions or produce unreliable results. Finally, cyber-attacks can pose a significant threat to autonomous systems, as they can manipulate sensor data or inject false information into the system.

To address these challenges, there is a need for resilient sensor fusion techniques that can ensure the reliability and robustness of autonomous systems. Resilient sensor fusion involves developing techniques that can detect and mitigate the impact of sensor failures, data uncertainties, and cyber-attacks. These techniques can include redundancy, fault detection, and correction mechanisms, data validation and filtering, and secure communication protocols.

This thesis proposes a framework for using existing sensors (such as range sensors, cameras, etc.) to integrate resilience. Sensor substitution is a key pillar of a "resilience feedback loop" where augmented information can be provided with alternative sensing sources. Furthermore, we will provide an overview of the challenges associated with sensor fusion in autonomous systems and the need for resilient sensor fusion. We will also discuss the various factors that can affect the reliability of autonomous systems and the importance of developing robust and resilient sensor fusion techniques. Finally, we will introduce the proposed methodology for resilient sensor fusion and describe how it addresses the challenges associated with autonomous systems.

1.3 Thesis objectives

The objective of this thesis is to investigate the challenges associated with multi-sensor fusion for autonomous localization and mapping, including sensor failures and data uncertainties. The thesis also aims to review the existing literature on multi-sensor fusion and explore the state-of-the-art techniques used for resilient sensor fusion in autonomous systems, while proposing a methodology that combines classical and deep learning techniques for multi-sensor fusion in autonomous localization and mapping, which is resilient to sensor failures and data uncertainties.

Furthermore, another goal here is to implement and evaluate the proposed methodology using simulation experiments and real-world data to demonstrate its effectiveness in improving the accuracy and robustness of autonomous localization and mapping. During the dissertation, the performance of the proposed methodology has been compared with existing multi-sensor fusion techniques for autonomous localization and mapping (SLAM), and the advantages and limitations of the proposed approach have been identified. The potential applications of the proposed methodology have been explored in various domains, including robotics, autonomous vehicles, and unmanned aerial vehicles.

Finally, another target of this thesis is to also provide recommendations for future research on multi-sensor fusion for autonomous localization and SLAM, and to identify potential areas for further improvement in the proposed methodology.

1.4 Motivation for research

In this section, a set of motivations for the research will be presented, highlighting their impact on the state of the art.

1.4.1 Integrate sensor resilience for existing hardware

One of the primary motivations for this research is to integrate sensor resilience techniques into existing hardware. Many existing autonomous systems rely on multi-sensor fusion techniques to provide accurate and reliable information about the environment. However, these systems may not be resilient to sensor failures and data uncertainties, which can have a significant impact on their performance and safety.

Integrating sensor resilience techniques into existing hardware can provide several benefits, such as improved reliability, increased robustness, and reduced downtime. By incorporating resilient multi-sensor fusion techniques, autonomous systems can continue to operate even when one or more sensors fail or provide unreliable data. This can improve the safety and efficiency of the system, reduce maintenance costs, and increase the lifespan of the hardware.

Furthermore, integrating sensor resilience techniques into existing hardware can be a cost-effective solution for many organizations that have already invested in autonomous systems. Instead of replacing the entire system, integrating resilience techniques can enhance the existing hardware's capabilities and improve its performance, without requiring significant investment.

The proposed methodology for resilient multi-sensor fusion in this thesis aims to integrate classical and deep learning techniques to improve the resilience of existing hardware for autonomous localization and mapping. This methodology combines redundancy, fault detection, and correction mechanisms with deep learning techniques to detect and mitigate the impact of sensor failures and data uncertainties.

Finally, we can improve the dependability, robustness, and performance of autonomous systems by incorporating sensor resilience approaches into current hardware. We can also provide enterprises that have already invested in autonomous systems a cost-effective solution.

1.4.2 Improve sensing ability for new and existing hardware

Another key motivation for this research is to improve the sensing ability of both new and existing hardware for autonomous tasks. With the increasing demand for more accurate and reliable autonomous systems, there is a need for more advanced sensing technologies that can provide a richer understanding of the environment.

Integrating resilient multi-sensor fusion techniques with classical and deep learning techniques can improve the sensing ability of both new and existing hardware. By combining multiple sensors, such as range sensors (UHF-RFID, UWB, etc.), IMU, LiDAR, camera,

GPS, and inertial sensors, autonomous systems can obtain a more comprehensive view of the environment and overcome the limitations of individual sensors. Additionally, by leveraging deep learning techniques, autonomous systems can extract more meaningful information from sensor data, such as object recognition and semantic segmentation.

Improving the sensing ability of autonomous systems can have a significant impact on various applications, such as robotics, autonomous vehicles, and unmanned aerial vehicles. For example, autonomous vehicles require highly accurate and reliable sensors to detect obstacles, navigate complex environments, and ensure passenger safety. Improving the sensing ability of autonomous vehicles can improve their performance, reduce the risk of accidents, and increase public trust in autonomous technology.

Moreover, improving the sensing ability of existing hardware can be a cost-effective solution for organizations that have already invested in autonomous systems. By integrating more advanced sensors and resilient multi-sensor fusion techniques, organizations can enhance the capabilities of their existing hardware and extend their lifespan.

By improving the sensing ability, we can provide a more comprehensive understanding of the environment, improve the accuracy and reliability of autonomous systems, and enable new applications in various domains.

1.4.3 Apply the resilient sensing framework to problems related to autonomous systems

Another motivation for this research is to apply the resilient sensing framework to problems related to autonomous systems, specifically to localization and Simultaneous Localization and Mapping (SLAM). Localization and SLAM are critical components of autonomous systems that enable them to navigate and map unknown environments. However, SLAM algorithms, in particular, are highly dependent on sensor data, which can be affected by sensor failures and data uncertainties.

Integrating resilient multi-sensor fusion techniques with classical and deep learning techniques can improve the accuracy and robustness of SLAM algorithms. By combining multiple sensors, autonomous systems can obtain more accurate and reliable data, which can improve the performance of SLAM algorithms. Additionally, by leveraging deep learning techniques, autonomous systems can extract more meaningful information from sensor data, such as object recognition, semantic segmentation and outlier detection, which can further improve the accuracy of SLAM.

Applying the resilient sensing framework to SLAM can have a significant impact on various applications, such as robotics, autonomous vehicles, and unmanned aerial vehicles.

For example, in robotics, resilient SLAM algorithms can improve the accuracy of robot navigation and enable robots to operate in more complex environments. In autonomous vehicles, resilient SLAM algorithms can improve the accuracy of vehicle localization, reduce the risk of accidents, and improve passenger safety. In unmanned aerial vehicles, resilient SLAM algorithms can enable more precise and reliable mapping of terrain, which can be useful for various applications, such as search and rescue, environmental monitoring, and agriculture.

Moreover, applying the resilient sensing framework to SLAM can provide a cost-effective solution for organizations that have already invested in autonomous systems. By integrating resilient multi-sensor fusion techniques with classical and deep learning techniques, organizations can improve the accuracy and robustness of their SLAM algorithms and extend the lifespan of their autonomous systems.

1.5 Contributions

The contributions of this thesis are the development of methodologies, algorithms, and test cases to provide several answers to the hypothesis using a multi-sensor approach that integrates sensor resilience, improve sensing ability and apply the resilient sensing framework to problems related to the autonomous systems. The resulting journal papers and peer-reviewed conference papers are presented in Chapter 6.

1. Sensor resilience for localization and SLAM [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]: Proposed approaches to localization and SLAM for autonomous systems (both unicycle-like robots and generic moving agents) based on classical and deep learning methodologies. The sensor inputs range from UHF-RFID to UWB range sensors fusing encoders mounted on the wheels and/or visual odometry coming from tracking cameras or stereo cameras; the designed systems integrate sensor resilience.

2. Improvement of sensing abilities [11], [12], [13]: Developed a methodology to increase the sensing abilities fusing several different visual odometry methodologies. The developed algorithms have been tested and applied on UAVs and UGVs. In the same context, a distributed architecture for UASs based on ROS 2 has been designed and proposed.

3. Sensor data processing and generation [14], [15], [16], [17], [18]): Developed several methodologies to process sensor data exploiting physical characteristics and phenomena (e.g. polarization mismatch for RFID). Design and development of methodologies

based on deep neural networks (DNN) to generate realistic sensor data to be exploited in the experimental setups.

1.6 Thesis outline

The thesis is organized with the following structure. Chapter 1 presents an introduction to the topics that will be presented in the thesis. Immediately following the introduction is an overview of the resilient robot perception topic and of the existing literature on resilient sensor fusion and the challenges associated with it in Chapter 2. Chapter 3 presents the proposed methodology for resilient sensor fusion, focusing on how the presented results have been achieved. Chapter 4 shows the framework for resilient multi-sensor fusion, and its application to heterogeneous sensors. Chapter 5 presents the experimental results and analysis of the proposed approach, showing the methodology effectiveness for several use cases. Chapter 6 shows the impact of research, analyzing the resulting publications. Chapter 7 reports the details about the impacts of the presented research on society. Finally, Chapter 8 concludes the thesis and outlines the future directions for research in the field of resilient sensor fusion for autonomous systems.

Chapter 2

Resilient Robot Perception

Establishing the spatial and temporal relationships between the robot and its surroundings is what perception is all about. In this chapter, we examine the most recent developments in the two key areas of the resilient robot perception problem: localization and SLAM. We explicitly introduce the resilient perception issue for intelligent robot systems first. Second, we concentrate on the localization and SLAM components, which are where our contributions are made. Third, we examine each module individually to provide the key connected works. Fourth, we outline the multi-sensor fusion problem and provide a summary of the current state of the art in relation to the most popular fusion approaches and the various fusion structures. Finally, we highlight the resilient fusion techniques and fusion architectures included in our contributions and conclude the examination of the state-of-the-art.

2.1 Robot perception problem

Environment perception for an autonomous robot refers to the ability of the robot to understand and interpret the environment in which it operates using various sensors and algorithms. The perception system of an autonomous robot typically consists of various sensors, such as cameras, LiDARs, radars, and sonars, that gather information about the robot's surroundings. The information collected by these sensors is then processed by algorithms that extract useful features and patterns, such as object detection, localization, mapping, and trajectory planning. Object detection algorithms enable the robot to identify and locate objects in its environment, such as obstacles, pedestrians, and other vehicles. Localization algorithms help the robot to determine its own position and orientation relative to the surrounding objects. Mapping algorithms enable the robot to create a map of its environment, which can be used for navigation and planning. Trajectory planning algorithms help the robot to plan its movements and avoid obstacles based on its perception of the environment. The environment

perception for an autonomous robot thus involves the use of sensors and algorithms to collect, process, and interpret information about the robot's surroundings, enabling it to navigate and interact with its environment in a safe and efficient manner. Environment perception provides, by processing sensor measurements, information about the environment the vehicle is immersed in. Perception module is the first of three main modules of an intelligent system and aims at modelling the environment. Reasoning & decision module uses the information obtained by the perception module to decide which actions are more adequate. Finally, the action module executes these actions. In order to obtain good reasoning and control we have to correctly model the surrounding environment. Figure 2.1 shows the interaction of the three main components of a generic autonomous robotic system. Establishing the spatial and temporal relationships between the robot, stationary objects, and moving objects in the scene constitutes perception. According to Wang's summary in [19], the vehicle must first be able

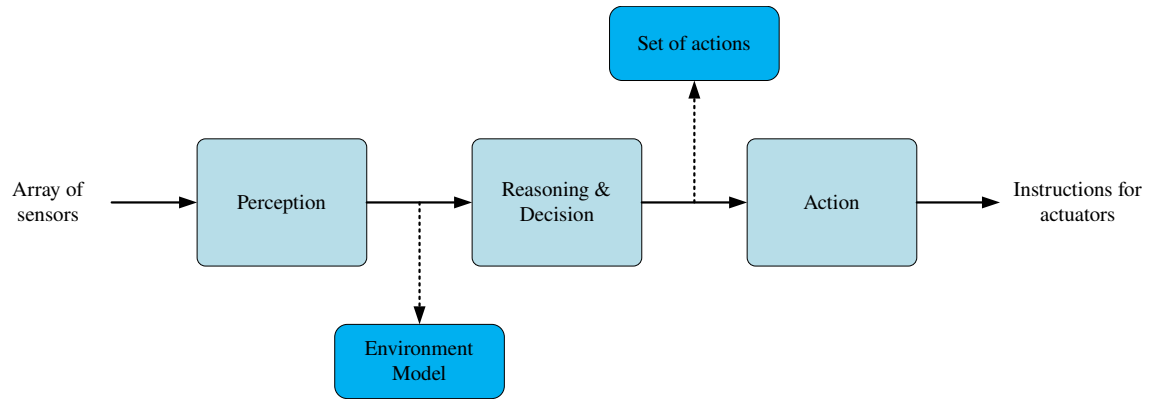


Fig. 2.1 Architecture of the perception module for an autonomous robotic system.

to locate itself in the scenario by establishing its spatial interactions with static objects in order to be able to sense the surroundings. Secondly, it has to build a map of the environment by establishing the spatial relationships among static objects. By establishing the spatial and temporal relationships between moving objects and the vehicle as well as between moving and static objects, it must then detect and track the moving objects. Perceiving the environment entails the selection of different sensors to gain a detailed description of the surroundings and an accurate identification of the things of interest. Due to the unpredictable and imprecise nature of the sensors data, all the processes engaged in the perception job are impacted, and have to manage uncertain inputs. Figure 2.2 shows an example of an environment representation for a real urban scenario, this figure shows the raw data provided by a lidar sensor and the final representation of the vehicle's surroundings obtained using this data. This representation should display the static and moving objects in the environment. Robot perception is based upon simultaneous localization and mapping (SLAM) that deals

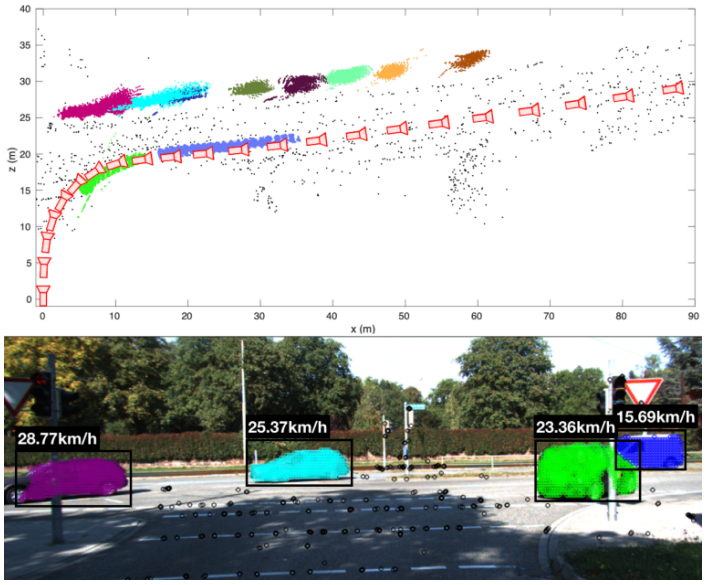


Fig. 2.2 Example of a robot perception system. In the top figure, the position of the robot moving in the environment is depicted (in red), together with the point clouds (in different colors) that represent the moving objects. In the bottom image, the frames acquired by a camera are depicted with the tracked moving objects with their estimated speed.

with modelling static parts. In SLAM, when the robot location and map are unknown the robot generates a map of the environment while simultaneously localizing itself within the map given all the measurements from its sensors. The final output is a model of the environment usually composed by the robot pose and the map of the static components. The management of incomplete information is an important requirement for perception systems. Incomplete information can be originated from sensor-related reasons, such as calibration issues, hardware malfunctions, miss detections, asynchronous scans; or from scene perturbations, like occlusions, weather issues and object shifting. These situations have to be managed properly in order to take into account the degree of imprecision and uncertainty and thus to provide an appropriate level of resilience.

Perception is, thus, the process of generating a representation of the environment surrounding the robot by interpreting data coming from robot sensors. The process involves the estimation of the robot pose in the environment and the relative positions of the objects around the robot. We exclude from this dissertation, the presence of moving objects. A graphical representation of the perception problem is then depicted in Figure 2.3. We need to formally define this problem, in order to discuss the state-of-the-art approaches to solve the perception problem. Given a set of sensor observations up to time t Z_t defined as:

$$Z_t = \{z_0, z_1, \ldots, z_t\}, \tag{2.1}$$

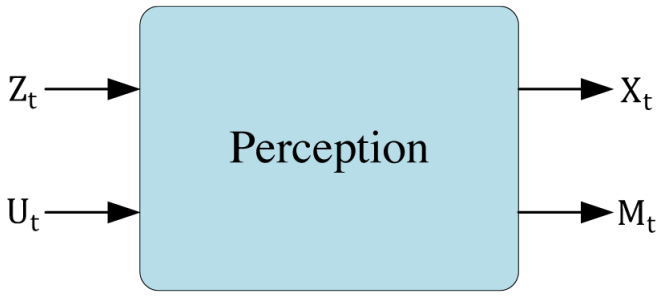


Fig. 2.3 Diagram for the intelligent robot perception.

and the control inputs up to time t U_t defined as:

$$U_t = \{u_0, u_1, \ldots, u_t\}, \tag{2.2}$$

we can define the perception problem as the ability to capture and represent the surrounding environment with respect to the robot pose. The robot perception provides, up to time t, the set of estimated robot states:

$$X_t = \{x_0, x_1, \ldots, x_t\}, \tag{2.3}$$

that, in general, are defined by its pose (position and orientation). Second, the perception module provides the set of static objects in the environment (i.e. the map) at time t:

$$M_t = \{m_{0,t}, m_{1,t}, \ldots, m_{K,t}\}, \tag{2.4}$$

where K is the total number of the objects in the environment and where each m_k, with $k \in [0, K]$, specifying the properties and the location of each object with respect to the robot.

The current literature allows us to define the perception problem as an *a posteriori* probability calculation:

$$P(x_t, M_t | Z_t, U_t, x_0), \tag{2.5}$$

where x_0 represents the initial state of the robot. The equation (2.5) can be regarded to as a SLAM problem, making the assumption that no moving obstacles have to be tracked or incorporated into the map M_t; these objects will be filtered by the SLAM algorithm not being part of the static map.

In the following sections, we describe in details the SLAM module.

2.1.1 Preprocessing

A sensor is a device that measures and converts a physical magnitude into readable data that can be processed by a computer. Sensors are the input mechanisms of perception systems and the only way to obtain direct information from the environment. Their counterparts are the actuators, which are the output mechanisms that interact with the environment. Usually, a sensor monitors a single feature of the environment; consequently, to collect information about all the essential characteristics of the environment, different sensors are often utilized. This allows the ability to obtain a more accurate and complete environment representation. Sensors can be divided into two groups: passive and active, depending on how they interact with their surroundings. On one hand, passive sensors only measure the energy emitted by entities in the environment. On the other hand, active sensors emit energy into the environment and then observe the change this energy causes in the state of the environment. How many and which sensors to use depends largely on the requirements of the application. For example, 2D laser scanners are the most popular choice when high accuracy of the position of an object of interest is required, while camera sensors may be a better choice when object detection is required. 3D laser scanners are becoming the most popular option today due to their high accuracy and ability to extract accurate volumetric information of objects from a scene. However, processing 3D data requires high computing resources. Also, the 3D laser scanner is not affordable for commercial applications.

2.1.2 Sensor uncertainty

We must be aware of the uncertain nature of sensors in order to accurately model their behavior. Hardware restrictions that only allow for the provision of an estimate of a real physical quantity may be the cause of sensor uncertainty. Environment noise or technological constraints and malfunctions of the sensors can cause random errors in the sensor results, and constitute another source of uncertainty. Uncertainty can emerge in a systematic fashion from calibration concerns and from transformation errors from the sensor space to a common representation ([20], [21]). Uncertainty management is an integral part of the perceptual task and involves the process of connecting uncertain and imprecise data with partially complete information and updating confidence coefficients. Sensor measurements are approximations of real physical quantities. Confidence in a measurement expresses a judgment about the validity of the information it provides. Its imprecision relates to its information content and its measurement value, while its uncertainty relates to the reliability of the measurement ([22]).

Sensor modeling is the description of probabilistic relationships between sensor measurements and the actual state of the environment, used to model its behavior [23]. In contrast to this concept, there are inverse sensor models that describe the probability of some environmental configurations in sensor measurements.

2.1.3 Localization

Localization is an important aspect of perception for autonomous mobile robots. For efficient navigation, robots need to adopt effective localization strategies. Here we want to present a comprehensive review on localization system, problems, principle and approaches for robots (see [24] for reference). We classify the localization problems in to three categories based on the information of initial position of the robot. Then, we discuss on robot position update principles and, finally, the techniques to localize the robot.

Localization principle

When moving through a given area, a mobile robot uses odometry to maintain track of its motion. The robot is unsure of its location due to odometry ambiguity. As a result, the robot must locate itself in relation to a map of its surroundings. Additionally, this prevents the location uncertainty from becoming limitless. The robot uses its exteroceptive sensors, including its laser, range, and ultrasonic sensors, to make observations about its surroundings in order to localize itself. Robot localization can be achieved by combining sensing data with odometry. Specifically, it is impossible to determine the robot's precise location even with a global tracking system (GPS). Only the best approximation of the robot's position can be determined from the data it can extract from its sensors. The belief of the robot is indicated by S_{bel}. The general process of robot's position update has two steps:

1. prediction/action update
2. perception/measurement/correction update.

The authors in [25] suggested to represent the beliefs about the robot pose as probability density functions (PDF). The prediction update is the phase where the robot uses its proprioceptive sensors (e.g. encoders mounted on the wheel, acceleration sensors, camera for visual odometry) to estimate its position. The odometry, however, brings errors and thus the uncertainty about the robot pose increases (see Figure 2.4 for reference). Assuming a single coordinate for the pose representation, the initial pose of the robot x_0 is known, thus the PDF is a Dirac delta function. During the movement, the robot pose uncertainty increases, due to odometric errors and it accumulates over time. During the perception (correction) phase,

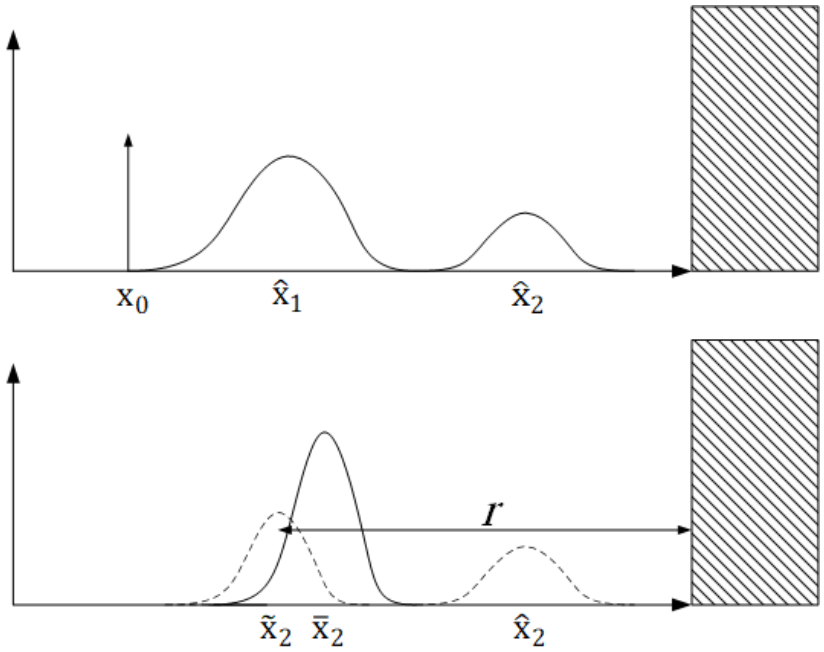


Fig. 2.4 Localization - Prediction and perception phases. Top: prediction phase. Bottom: perception phase.

the robot corrects the estimated position using the onboard exteroceptive sensors during the perception update phase. As an example of exteroceptive sensor, here we suppose that the robot uses a range sensor (e.g. UWB, UHF-RFID, laser scanner) to compute its current distance r from the right wall and then its current pose $\tilde{x}_2$; this position, however, conflicts with the position $\hat{x}_2$ estimated in the correction phase. The measurement update corrects the new location to $\bar{x}_2$, consequently the uncertainty shrinks (solid line in Figure 2.4).

We can define the robot pose at discrete time k, for a differential-drive kinematics, as follows:

$$p_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}. \tag{2.6}$$

We can then write the discrete time dynamics of the robot as:

$$p_{k+1} = \begin{bmatrix} x_k + \frac{u_{R,k}+u_{L,k}}{2}\cos(\theta_k) \\ y_k + \frac{u_{R,k}+u_{L,k}}{2}\sin(\theta_k) \\ \theta_k + \frac{u_{R,k}-u_{L,k}}{d} \end{bmatrix}, \tag{2.7}$$

with $u_{R,k}$ and $u_{L,k}$ representing the distance covered in the interval $(k\delta_t,(k+1)\delta_t)$ (i.e. at time step k, with discretization step δ_t) by the right and left wheels, respectively and d is the distance between the two wheels. The distance $u_{R,k}$ covered at time step k by the right wheel is related to a noisy encoder reading $u^e_{R,k}$ by the relation $u^e_{R,k} = u_{R,k} + n_{R,k}$, where the noise term $n_{R,k}$ is assumed a 0-mean Gaussian random variable with variance given by $K_R|u^e_{R,k}|$, being K_R a positive constant. A similar argument can be applied to the left wheel. In order to simplify the notation, the following definitions can be adopted:

$$\begin{aligned} u_k &= \tfrac{u_{R,k}+u_{L,k}}{2}, \quad u^e_k = \tfrac{u^e_{R,k}+u^e_{L,k}}{2}, \\ \omega_k &= \tfrac{u_{R,k}-u_{L,k}}{d}, \quad \omega^e_k = \tfrac{u^e_{R,k}-u^e_{L,k}}{d}. \end{aligned} \tag{2.8}$$

Thus, Equation (2.7) can be simplified as follows:

$$p_{k+1} = \begin{bmatrix} x_k + u_k\cos(\theta_k) \\ y_k + u_k\sin(\theta_k) \\ \theta_k + \omega_k \end{bmatrix}, \tag{2.9}$$

Localization approaches

In this section mobile robot localization approaches will be analyzed from a probabilistic perspective. We will present Markov localization, Kalman filter (KF) and evolutionary approaches.

Probabilistic approaches Probabilistic localization approaches based on map identify the probabilities of a robot being in specific positions. The errors in measurement affect sensor data, therefore, the probability of a robot in a specific pose can only be computed.

Markov localization The robot can locate itself starting from an unknown position. Multiple possible positions can be tracked by the robot such that Markov localization can recover from ambiguous situations. However, the state-space needs to be represented in a discrete way for updating the probability of possible positions. This discrete representation may be a topological graph or a geometric grid. The memory requirement for the map size is limited. In the prediction update phase, the robot's current location is estimated depending on the available information on previous locations and odometry input. The current state of the robot $\bar{S}_{bel}(p_k)$ can be computed from the previous estimated position $\bar{S}_{bel}(p_{k-1})$ and

control input (proprioceptive data) u_k:

$$\bar{S}_{bel}(p_k) = \sum_{p_{k-1}} P(p_k|u_k, p_{k-1})\, S_{bel}(p_{k-1}). \tag{2.10}$$

Equation (2.10) represents the prior belief of the robot's state. In the perception (update) phase, the robot corrects its previous position by combining it with the exteroceptive sensor inputs. The Bayes rule can then be applied to compute the new robot state $S_{bel}(p_k)$ as a function of its measurement data z_k and the previous state $\bar{S}_{bel}(p_k)$:

$$S_{bel}(p_k) \propto P(z_k|p_k, M)\, \bar{S}_{bel}(p_k), \tag{2.11}$$

where $P(z_k|p_k, M)$ denotes the probabilistic measurement model, namely the probability of observing z_k, given the knowledge of robot pose p_k and map M. Equation (2.11) represents the posterior belief of the robot's state. The probabilistic measurement model is calculated from a noise-free measurement function h which depends on M and p_k. In order to derive the probabilistic measurement model, a noise term is added to the measurement function in such a way that the probabilistic distribution $P(z_k|p_k, M)$ peaks at $h(p_k, M)$ which is noise-free. Assuming a Gaussian noise, we have:

$$P(z_k|p_k, M) = \mathcal{N}(h(p_k, M), R_k), \tag{2.12}$$

where $\mathcal{N}$ denotes a multivariate normal distribution having mean $h(p_k, M)$ and noise covariant matrix R_k.

Kalman filter-based localization The Kalman filter-based localization solves the position tracking problem in efficient way. This is an optimal sensor fusion approach (for linear systems with Gaussian noise) which tracks the robot from an initially known location. Usually, this localization can be used in continuous world representation. However, in some situations, the robot's uncertainty is too large and therefore not really unimodal. In this case the localization approach can't capture the possible robot positions and becomes lost irrevocably. Kalman filter is a special case of Markov localization, but it does not use arbitrary density function and, instead, it uses Gaussians to represent the robot's belief $S_{bel}(p_k)$, the motion model, and the measurement model. A Kalman filter is divided into different phases: the first phase (i.e. the prediction update) directly applies a motion model having Gaussian error to the robot's measured encoder travel. The perception update phase can be divided into the following steps:

- Observation step. The robot extracts different features from the sensor data.

- Measurement prediction. The robot generates a measurement prediction consisting of features that it expects to observe from its estimated position.
- Matching. The robot computes the best match between the features extracted from the observations and the expected features.
- Estimation. The robot belief state is updated fusing the matching information.

For a multi-variate Gaussian distribution, the multi-variate normal PDF can be defined as:

$$P(\mathbf{x}) = (2\pi)^{-n/2} \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right) \tag{2.13}$$

where μ is the mean vector, Σ is the covariance matrix (symmetric and positive semi-definite matrix) and $\mathbf{x}$ is an n-dimensional vector. In Kalman filter localization, the Gaussian is defined by mean μ_k and covariance Σ_k; these parameters are updated during prediction and measurement phase.

The Kalman filter approach to localization is very efficient, if compared to Markov localization, however, the robot initial position should be known with some approximation and, if the robot gets lost, it cannot recover its position (in contrast to Markov approach). Therefore, the position tracking problem is effectively solved by Kalman filter, but it does not solve the global localization problem efficiently. Another assumption in the theory of Kalman filter is that the system is supposed to be linear and with Gaussian noise. However, many robotic systems have nonlinearities. In order to overcome this problem, an extension of the KF to non-linear systems has been developed (Extended Kalman Filter, EKF). In the EKF, the system is linearized and then the standard KF is applied.

There are other probabilistic localization techniques that have been used in mobile robot research platforms such as unscented Kalman filter (UKF), grid-based localization and Monte-Carlo localization (MCL). UKF, like the EKF, also assumes Gaussian distributions. However, the UKF applies unscented transform to propagate and update the state and its uncertainty through the non-linear models accurately. In contrast, grid-based localization and MCL are not bounded by unimodal distributions. In grid localization, the robot belief is represented by a histogram filter. The MCL uses particle filters to represent the robot belief; it uses a subset of the whole set of possible positions to construct the approximate belief state for the robot. This results in tracking and updating a small number of possible locations instead of all possible locations which in turn reduces the complexity.

Evolutionary approaches Different authors applied evolutionary approaches such as particle swarm optimization (PSO) ([26] and [27]), genetic algorithm (GA) ([28] and

[29]) and differential evolution (DE) to basic localization techniques like EKF, MCL, PF etc. for determining the robot location. The authors in [26] proposed a PSO based localization technique which overcomes several issues found in PF based schemes. Localization using PF has major issues such as degeneracy and impoverishment problem which reduces its performance and PF approaches are also computationally demanding. The proposed method establishes a recursive framework for tracing out the robot position. This is achieved by converting the robot localization to dynamic optimization. The developed scheme neither requires resampling phase as in PF based methods nor distribution of noise. To estimate the robot position, a stochastic search technique is implemented in the state-space. The proposed PSO based approach is proved to be more efficient in comparison with standard PF and EKF based localization techniques. The authors in [30] presented a hybrid localization approach which combines DE with PSO. The proposed methods are compared with standard MCL and proved to give better result because of faster convergence and robustness. In [31], the authors proposed a DEPSO algorithm (DE + PSO). The first phase utilizes the selection and mutation operations of standard DE approach and then swarms best position is updated by PSO. The next phase updates velocity and position of particles using PSO followed by crossover and selection by DE. The two challenging criteria such as global optimization (of DE) and fast convergence (of PSO) makes DEPSO robust and powerful. Furthermore, in [32] the authors proposed an effective multi swarm PF for robot positioning. The major focus of this approach is increasing the performance of PF by PSO. This overcomes the localization impoverishment occurred in PF based scheme. In [33] the authors combined PSO with standard EKF for estimating the robot's current position. The EKF combines map-matching, position and orientation estimation and dead-reckoning approach. This result in reliable position tracking. However, in case of high error in map-matching, the proposed localization system applies PSO to relocate the robot. This is because, unreliable map-matching makes the robot go missing. The large map-matching error in the present estimate outcomes in low confidence. In this situation, PSO globally localizes the robot. Therefore, over the global map, a set of particles are published to search the room with high confidence to discover an outcome. In this paper, the estimation of pose is obtained by using the RPROP (Resilient Propagation) for error function minimization. The proposed system analyses the pose error in virtual and realistic scenario. In [27] the authors developed a self-localization approach for localizing soccer humanoid robots using image processing. It converts the obtained image into an image taken from the top view by an inverse perspective map. Then it utilizes PSO to define the robot's location relative to origin. The proposed method recognizes position based on captured images. The method achieves high accuracy as it utilizes ground lines points for self-localization. With some set of points this method can recognize its own position.

This causes it to be very noise resistant. The authors in [34] combined PSO and ant colony optimization (ACO) to estimate the robot location. PSO is used for weight adjustment and ACO utilizes the neural network topology. Finally, the authors in [28] proposed GA-fuzzy logic controller (FLC) to adjust the covariance matrices. GA makes the system more powerful because it tunes the fuzzy membership functions (MF) which results in improving FLC's accuracy. Finally, the comparison among EKF, fuzzy logic (FL)-based EKF and EKF based on GA-FL proves that GA-FL based EKF technique provides accurate result in comparison other two approaches.

2.1.4 Simultaneous Localization And Mapping

Simultaneous localization and mapping (SLAM) allows robots to operate in an unfamiliar environment and incrementally build a map of the environment using information provided by sensors. At the same time, the robot uses this map to determine its position. This map should only consist of static elements of the environment. This so-called static map needs to detect and exclude moving objects. Accurate location information is required to detect and track moving objects near moving vehicles. Location-based sensors such as GPS and DGPS often fail in unclear areas.

In order to perceive the surrounding environment, we must generate a map representation of the environment. Three main approaches are widely used by current state of the art mapping applications:

- Direct approach uses raw data measurements to represent the physical environment without extracting predefined features (e.g. point clouds representations) as in [35]
- Feature-based approach compresses measurement data into predefined features, for example geometric primitives like points, lines, circles as in [36]
- Grid-based approach subdivides the environment into a regular grid of cells, and then a probabilistic measure of occupancy is estimated for each cell as in [37] and in [38].

Despite the advantages of representing any type of environment, a direct approach to memory usage is impractical and cannot represent sensor uncertainty. The feature-based approach is compact, but simple primitives fail to adequately represent complex environments. A grid-based approach is better suited for this task as it can represent arbitrary features, provide a detailed representation, and consider sensor properties through the definition of the sensor model. Figure 2.5 shows an example map representation obtained by his three main approaches above. The occupancy grid approach has become the most common choice among map representation methods for outdoor environments due to its advantages over the

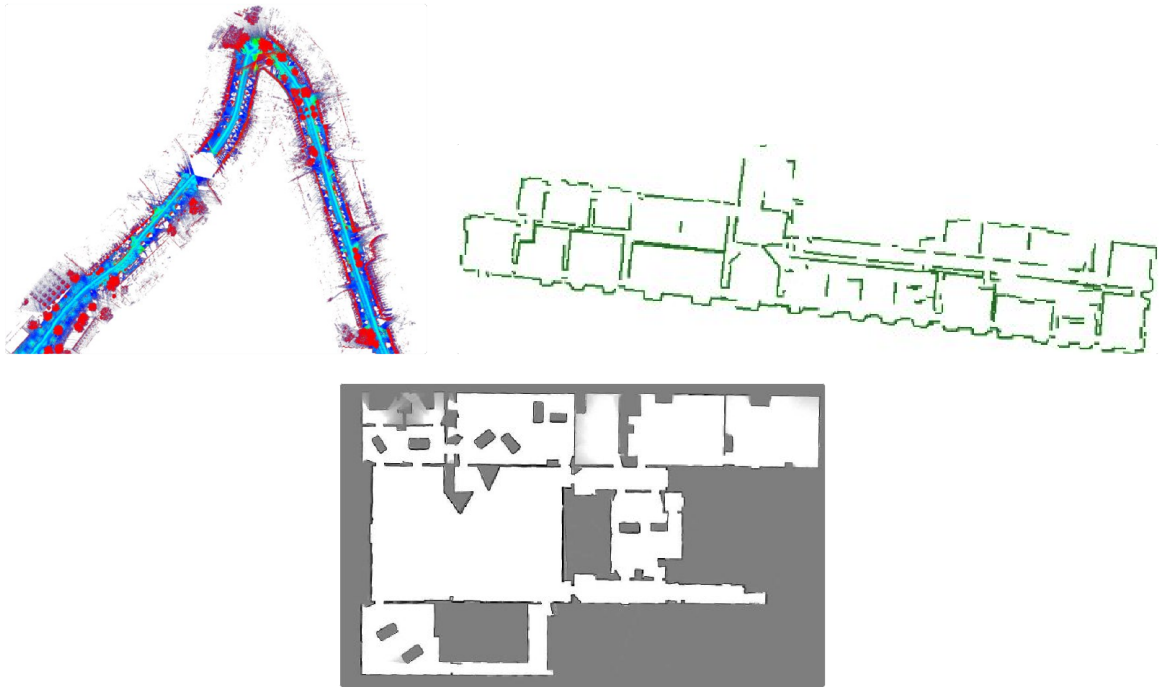


Fig. 2.5 Example of map representations. The top left image is a point cloud map, the top right is a feature map, whilst the bottom is an occupancy grid map.

others. Its main drawback is the amount of memory needed to represent a large outdoor environment; however, in order to overcome this issue, the works presented by [19] and [39] help us with an idea that since the range of the sensors is limited, there is the only need to construct a local grid map limited by the non-measurable regions. Afterwards local maps are assembled to build a global map. [39] proposes a good solution to solve the SLAM problem which is based on an occupancy grid to represent the robot map, and free-form objects to represent moving entities. To correct robot location from odometry he introduces a new fast incremental scan matching method that works reliably in dynamic outdoor environments. After a good robot location is estimated, the surrounding map is updated incrementally. The results of his method in real scenarios are promising.

Robotic perception heavily relies on knowing the location or localization of an autonomous robot. Imagine placing a robot in a strange setting. The robot must be able to map the environment using sensor inputs, but in order to do so, it must be aware of its location inside the uncharted territory and vice versa. The contradiction within the issue makes it difficult to solve: in order to move accurately, a robot requires an accurate map of its surroundings, yet in order to create an accurate map, the robot has to be aware of its exact location within the map. SLAM is one of the most challenging task in robotics because of the close relationship between these two issues. Simultaneous Localization and Mapping

(SLAM) is a restatement of the prior issues: a robot must be able to incrementally create a consistent map of the environment using sensor inputs while simultaneously determining its location within the map when it is placed in an unknown location in an unknown environment. In a SLAM solution, the robot builds a map of its surroundings and concurrently localizes itself there using all data from input sensors and odometry. (e.g. camera, radar, lidar).

Initial solutions to solve the SLAM problem can be traced back to the works in [40] and in [36]. In these publications, the authors use the manipulation of geometric uncertainty and measurement uncertainty to create a statistical framework for expressing uncertain connections between landmarks. Additionally, these studies found that estimations of landmarks viewed by the vehicle are associated as a result of the usual mistake in estimating the location of the vehicle. As a result, the locations of each landmark were added to the state vector and updated after every observation. The robot's requirement for computation and storage rose as it searched the unfamiliar area for new landmarks. Many solutions create thresholds or do not take into account correlations between landmarks in order to restrict the resources expansion ([41]). These works are considered as the basis of modern and more complex SLAM techniques.

Initial SLAM systems were powered by landmarks, which served as the primary feature utilized to represent the surroundings. However, depending just on landmarks to find a vibrant outside setting turned out to be insufficient. The occupancy grids (OG) architecture for localization and mapping was proposed in [23]. This framework discretizes the real world into cells, each of which is assigned a probabilistic estimate of its occupancy status. Typically, a high cell probability value denotes an occupied cell, whereas a low one denotes a free or empty cell. By using Bayesian approaches, occupancy levels are continuously updated with new sensor measurements. It is obvious that the grid resolution and processing and storage capabilities are closely connected. For real-time applications, it is feasible to achieve a trade-off between representation power and time processing, as demonstrated by a number of techniques ([42], [39] and [19]). Studies like those in [43] and in [44] demonstrate that Dempster-Shafer theory ([45]) may be used to describe occupancy when taking into account evidence distributions that can represent one occupancy state without making assumptions about the others. These efforts provide an intriguing way to distinguish between the unknown (no information) and uncertainty brought on by contradictory data concerning cell occupancy that has been steadily obtained. Additionally, occupancy grid techniques create a standard representation that may be applied to sensor fusion.

The probabilistic SLAM presented in [39] and in [19] is a widely applied SLAM solution. This is based on the Bayes' rule and Markovian assumptions and requires the estimation of

the following probability distribution:

$$P(x_t, M_t | z_{0:t}, u_{0:t}) \propto P(z_t | x_t, M_t) \int P(x_t | x_{t-1}, u_t) P(x_{t-1}, M_t | z_{0:t-1}, u_{0:t-1}) dx_{t-1}, \quad (2.14)$$

where x_t is the robot's pose at time t, $z_{0:t}$ are the measurements obtained by the robot up to time t, $u_{0:t}$ are the control inputs applied to the robot up to time t, and M_t is the map of the environment at time t. The equation (2.14) is composed of two recursive steps: prediction and correction. The prediction step performs an update based upon:

$$P(x_t, M_t | z_{0:t}, u_{1:t}) = \int P(x_t | x_{t-1}, u_t) P(x_{t-1}, M_t | z_{0:t-1}, u_{0:t-1}) dx_{t-1}, \quad (2.15)$$

where the factor $P(x_t | x_{t-1}, u_t)$ is known as the robot model. The correction step updates the information, based on the new available measurements:

$$P(x_t, M_t | z_{0:t}, u_{1:t}) = \frac{P(z_t | x_t, M_t) P(x_t, M_t | z_{0:t}, u_{0:t})}{P(z_t | z_{0:t-1}, u_{0:t})}, \quad (2.16)$$

where the factor $P(z_t | x_t, M_t)$ is known as the sensor model. The robot model indicates how the vehicle has moved from its previous to current state according to the control inputs (e.g. control commands or odometry values), while the sensor model defines the probabilistic description of the sensors used to perceive the environment.

Map representation

A crucial decision that serves as the foundation for subsequent perception tasks is how to depict the environment on the map. Data compression, the degree of environment representation, uncertainty management, and sensor settings all play a role in this choice, which is application-specific. The most common representation techniques are listed after these elements and the categorization offered in [39] as follows:

- **Direct representation**. The map is created using raw sensor data. Due to the utilization of points (mostly range sensors) as the primary components for depiction, this technique is often referred to as a "point cloud." The approach is simple and has a high degree of representation, but it lacks uncertainty management and has a high computational cost. This form is used in [46] and in [47] for SLAM in 3D settings and mapping utilizing range sensor scans, respectively.

- **Feature representation**. To discover and extract specific characteristics that are utilized to describe the map, raw sensor data is processed. The depiction is made

Table 2.1 Feature comparison among the main map representations schemes.

Scheme	Data compression	Detailed representation	Uncertainty management	Sensor features
Direct		X		
Feature-based	X		X	
Grid-based		X	X	X

more condensed by the use of features. The set and quantity of characteristics have a significant impact on the quality of representation. Although geometrical elements are desirable, they perform less well in irregular situations where the variety of forms provide a problem. In order to describe the environment, [36] and [48] presented SLAM methods employing a variety of geometrical forms and landmarks.

- **Grid-based representation**. This representation, which was first presented in [23], discretizes the environment into regular cells and derives the occupancy status of each cell from the sensor readings taken at each instant. For applications involving both indoor and outdoor perception, this strategy is common. The grid resolution has a significant impact on the degree of representation and data reduction.

We consider feature representation as the best suited for our proposed approach. Grid-based representation may require high amounts of storage (lack of data compression) and computing processing, for that reason this is not the best suited method to represent the map. The feature-based representation has the ability to manage uncertainty from sensor measures and include data compression as shown in Table 2.1.

2.1.5 Visual Simultaneous Localization And Mapping

Visual simultaneous localization and mapping (vSLAM) is a technique that combines computer vision and robotics to enable a robot to navigate an unknown environment while simultaneously building a map of it. Like the previously discussed technique of simultaneous localization and mapping (SLAM), vSLAM is an important tool for robots that operate in environments where accurate maps are not available. While traditional SLAM techniques rely on sensors such as sonar, lidar, or depth cameras, vSLAM uses visual sensors, such as cameras, to build a map of the environment. By analyzing the images captured by the camera, a robot can identify and track features in the environment, and use this information to determine its position relative to the map.

In many ways, vSLAM builds on the fundamental concepts of SLAM. Both techniques rely on the robot's ability to track its position and movement, and to use sensor data to update its map of the environment. However, vSLAM introduces a new set of challenges and opportunities. One major advantage of vSLAM is that it can operate in environments where other sensors may not be effective, such as in low-light conditions or when there is interference from other sources. Additionally, visual data can provide a richer and more detailed map of the environment, which can be useful in applications such as augmented reality and navigation for autonomous vehicles. However, vSLAM also presents unique challenges. For example, visual sensors can be prone to errors such as motion blur, lens distortion, and changes in lighting conditions. Additionally, processing and analyzing large amounts of visual data in real-time can be computationally expensive. Overall, vSLAM is a rapidly evolving field that has the potential to revolutionize robotics and computer vision. By combining the strengths of both disciplines, vSLAM can provide robots with the ability to navigate and map the world around them, opening up new possibilities for applications in a wide range of industries.

The three primary categories of visual-based techniques are visual-only SLAM, visual-inertial (VI) SLAM, and RGB-D SLAM. The first one relates to SLAM methods that solely use stereo or monocular cameras to capture 2D pictures. Due of their limited visual input, they provide a significant technological challenge [49]. By incorporating an inertial measurement unit (IMU), which is available in their miniaturized size and low cost while achieving high accuracy, essential aspects of many applications that require lightweight design, such as autonomous race cars [50], the robustness in the sensor's tracking of the visual-SLAM algorithms may be increased. A depth sensor may also be used by visual-based SLAM systems, and the depth data may then be processed using an RGB-D method.

The work in [51] and in [52] suggests a SLAM approach that goes from the issue definition to the environment models in order to get a broad summary and introduction to the SLAM problem. Additionally, in [53] the authors evaluate the primary unresolved issues and potential developments for the SLAM. The authors in [54] and in [55] offer an overview of the key concepts used in the visual-only SLAM techniques and the basic algorithms taking into account reviews and polls of visual-based techniques. The RGB-D-based SLAM issue is also briefly discussed in [54]. The major ideas used in the visual-based SLAM approaches are presented in an overview in [56] and in [49], with an emphasis on the visual-only and RGB-D-based approaches and descriptions of the key algorithms. In their latest article [57] the authors suggest a categorization of the primary visual-based SLAM algorithms and conduct a historical study. The basic ideas and methods of the visual-inertial SLAM and visual-inertial odometry approaches are presented by the authors in [58] and in [59], taking

into account the filtering-based and optimization-based viewpoints. The methods used up until 2015 are given in [58], and the formulas used up until 2018 are also included in [59]. [60] also provides a summary of the key ideas and methods used in visual-inertial guidance. The authors in [61] give a global view from the primary ideas used in RGB-D modeling with regards to the RGB-D approaches. An summary of the key ideas and an explanation of the major RGB-D-based SLAM algorithms are presented in a recent study in [62]. There are a number of studies and surveys about visual-based SLAM methods in the literature; however, the majority of them are restricted to just one or two of the three primary approaches and do not go into depth about the algorithms. A review that covers the three approaches and the basic algorithms as also presented in [63] is then given here.

Visual-Based SLAM concepts

The major traits of the visual-based approaches are discussed in this part, along with concepts linked to visual-based SLAM and odometry algorithms. One or more sensors in the sensor system serve as the information source for the visual-based SLAM methods, which receive 2D pictures. Initialization, tracking, and mapping are the three major components that make up the visual-based SLAM algorithms [49]. Figure 2.6 depicts the three major components that are typically present in methods to visual-based SLAM. In Figure 2.6, depending on the

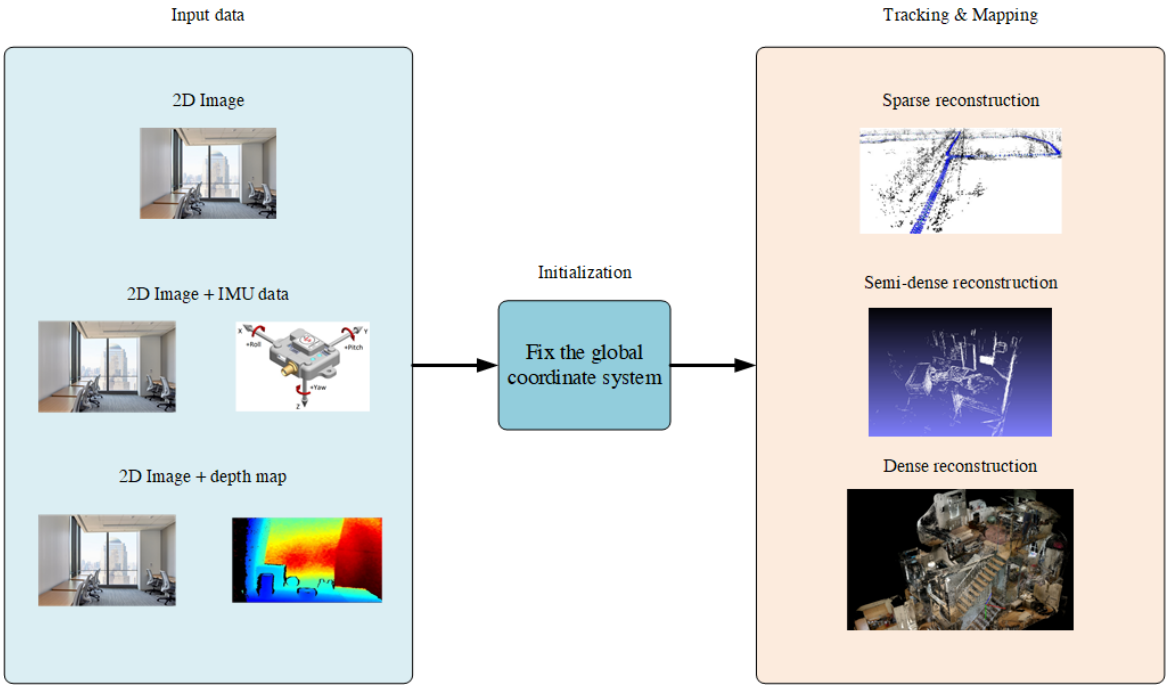


Fig. 2.6 General components of a visual-based SLAM. The depth and inertial data may be added to the 2D visual input to generate a sparse map, semi-dense map and a dense reconstruction.

used approach—namely, visual-only, visual-inertial, or RGB-D-based—the input for visual-SLAM systems can be a 2D picture, a 2D image and IMU data, or a 2D image and depth data. The two primary stages of monitoring and mapping are determined by the initialization, which also creates an initial map. The constant assessment of the sensor's position is carried out by the tracking procedure. The algorithm, which solves the perspective-n-points issue, creates 2D–3D correspondences between the present frame and the map. This issue can be resolved in a number of methods, with EPnP (enhanced Perspective-n-Point) being one of the most prevalent approaches [64]. The 3D framework is computed and expanded as the camera travels thanks to the mapping process. The utilized method affects how the depth data is calculated. Finally, depending on the method used, the mapping steps should produce a sparse, semi-dense, or dense 3D reconstruction. Despite the fact that we primarily refer to the ideas as being part of the SLAM methodology, we also take into account visual-SLAM and visual-odometry (VO) techniques in this article because of their close connections. Through sensors as a source of data, the VO algorithms also attempt to predict a robot's location. The primary distinction between visual-SLAM and VO is whether or not the predicted route and map are taken into account globally [54]. While VO only conducts local optimizations, loop closure detection is a feature of visual-SLAM algorithms, allowing them to rectify drifts that have gathered at the conclusion of the robot's trajectory.

Visual-Only SLAM The foundation of the visual-only SLAM devices is 2D image analysis. The system conducts the initialization process to establish a global coordinate system and rebuild an initial map following the collection of pictures from multiple points of view. Initializing the map points with high ambiguity in the feature-based algorithms that depend on filters (filtering-based algorithms) is the first stage. These map points may later resolve to their real locations. Following this step is tracking, which makes an effort to determine the camera's position. As more unfamiliar events are witnessed, the mapping procedure simultaneously adds new points to the 3D reconstruction.

The binocular or stereo camera used by the visual-only SLAM technology is an option. Given the tiny size of the sensor (the smallest of all the offered approaches), its cheap cost, simple calibration, and its low power usage, the monocular camera-based SLAM is a well-explored area [65]. Despite these benefits, monocular-based systems have a greater initialization complexity because at least two distinct perspectives are required to identify the initial depth. They also present estimation issues with drift and scale estimation. Stereo cameras, which offer the primary benefit of featuring the stereo image in just one frame, can help to solve this last issue. However, compared to a straightforward binocular camera, the sensor area is more important. Additionally, each frame needs to go through more processing

because of the need for picture correction during the stereo matching step. There are two primary categories of visual-only SLAM: feature-based and direct.

Direct methods The direct techniques, in contrast to the feature-based ones, use the sensor data directly, without any pre-processing, while also taking into account the intensities of the pixels and reducing photometric error. This approach has a wide range of methods, and the reconstruction can be dense, semi-dense, or sparse depending on the strategy used. Since the combined optimization of both structure and camera locations is computationally more costly for dense and semi-dense reconstructions than for a sparse one, the reconstruction density is a significant barrier to the algorithm's real-time operation [66]. According to their front-end and back-end, or the portion of the algorithm responsible for the abstraction of the sensor's data and the portion responsible for the analysis of the abstraction, respectively.

Feature-based methods Features-based SLAM methods take into account a predetermined number of keypoints, also known as points of interest. They can be found in multiple pictures and matched by contrasting their characteristics; this procedure yields information on camera pose prediction. The feature, or the data used by the algorithm to perform the tracking and mapping, is made up of the descriptor data and keypoint position. The feature-based techniques can be used in embedded versions because they do not utilize all of the frame information. The feature extraction, however, might not succeed in a textureless world [67] and produces a sparse image that offers less information than a dense one. Figure 2.7 illustrates the primary distinction between feature-based (indirect) and direct methods.

Visual-Inertial SLAM Inertial readings are used in the VI-SLAM method to calculate the structure and sensor position. An inertial measurement unit (IMU), which combines a gyroscope, an accelerometer, and a magnetic instrument, is used to collect the inertial data. In this manner, the IMU is able to provide data on the angular rate (gyroscope), acceleration (accelerometer), and, additionally, the magnetic field surrounding the instrument (magnetometer). While adding an IMU may increase the environment's information richness and offer greater accuracy, it also makes the algorithm more complex, especially during the initialization step because the algorithm must also estimate the poses of the IMU in addition to the camera. The sort of fusion between the camera and IMU data, which can be loosely or closely linked, can be used to categorize VI-SLAM methods. The IMU data are used to estimate the direction and shifts in the sensor's location by the weakly linked techniques, which do not combine the IMU states to estimate the complete posture [59]. The closely coupled techniques, on the other hand, rely on fusing camera and IMU data into

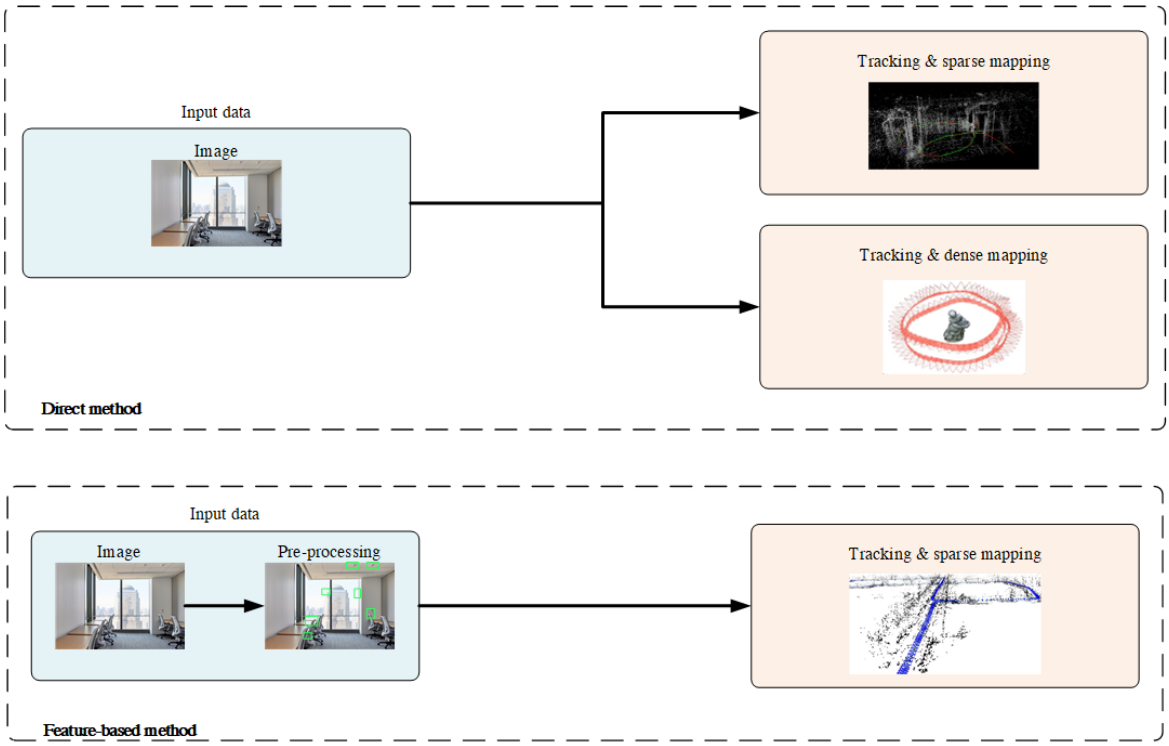


Fig. 2.7 General differences between direct and feature-based methods. Top: main steps followed by a direct method, resulting in a sparse (generated from the reconstruction of *sequence_02/TUM* MonoVO [68] with the DSO algorithm [69]) or dense reconstruction (Reprinted from [70]), according to the chosen technique. Bottom: main steps followed by the feature-based methods, resulting in a sparse reconstruction (map generated with the ORB-SLAM algorithm ([71], [72]) in the *MH_01/EuRoC* sequence [73]).

a motion equation to produce a state prediction that takes into account both types of data. Additionally, VI-SLAM algorithms offer various versions based on their back-end strategy, which may be optimization- or filtering-based. While optimization-based methods (also referred to as keyframe-based approaches) depend on global optimizations, which boost the system's precision as well as the algorithm's computational cost, filtering-based approaches for VI-SLAM rely on feature extraction at the front end.

RGB-D SLAM With the release of Microsoft's Kinect in 2010, SLAM systems built on RGB-D data began to gain more notice. RGB-D sensors enable SLAM systems to directly obtain the depth information with a practicable precision achieved in real-time by inexpensive hardware. They are composed of a monocular RGB camera and a depth sensor. Since the RGB-D devices immediately supply the depth map to the SLAM systems, this approach's basic framework for SLAM is different from the others that have already been demonstrated. The majority of RGB-D-based systems use the depth maps and iterative closest point (ICP) method to find the sensor and reconstruct the entire structure. The benefits of RGB-D systems include the provision of dense depth maps and color picture data without the need for pre-processing, which reduces the difficulty of SLAM setup [49]. Despite this, this strategy works best [74].

Visual-SLAM algorithms

The previous section's examined approaches each include a number of algorithms, making it challenging to choose the best SLAM or odometry method for a given project's requirements. In order to achieve a short overview of each strategy and a systematic analysis based on six chosen criteria that, in general, are presented as limiting factors of SLAM projects, we present the most typical algorithms of each approach, picked based on literature input. In addition to the suggested criteria, it is important to describe the scene and application because some situations may have characteristics that call for particular assessment criteria, as in the analysis in [75]. Six parameters that affect system dimensioning, accuracy, and hardware implementation taking into account the general SLAM system strategy can be introduced. The following factors are important to consider: algorithm type, map density, global optimization, loop closure, availability, and embedded implementations.

- Algorithm type: This criterion describes the approach that the program has used. We categorize the visual-only algorithms into feature-based, mixed, and straight approaches. They must be founded on filtering or optimization, given the visual-inertial techniques. Finally, there are three different types of monitoring methods for the RGB-D approach: direct, mixed, and feature-based.

- Map density: Generally speaking, a dense reconstruction uses more processing resources than a sparse one, affecting memory utilization and computational cost. On the other hand, it offers a more precise and thorough reconstruction, which could be crucial for SLAM projects.
- Global optimization: Global map optimization, which is a method for compensating the accumulated mistake brought on by camera movement while taking the consistency of the complete building into account, may be included in SLAM algorithms.
- Loop closure: The SLAM algorithm's ability to recognize previously detected images in order to calculate and rectify the drift collected during sensor movement is referred to as the loop closure detection.
- Availability: A number of SLAM algorithms are open source and made accessible by the writers or have third parties make their implementations available, making it easier for people to use and reproduce them.
- Embedded implementations: Embedded SLAM implementation is a developing area with many uses, particularly in the automation and automotive industries. Since there must be a trade-off between algorithm design in terms of energy consumption, memory usage, and processor usage, this measure relies on the hardware limitations and uniqueness for each algorithm. We compiled the key articles we discovered showcasing completely integrated SLAM systems in hardware like microcontrollers and FPGA boards.

In the present dissertation we focus on Visual-Only SLAM algorithms that have been reported on a timeline in Figure 2.8 and specifically on the ORB-SLAM2 algorithm [72] that has been widely used in literature and has very good performances.

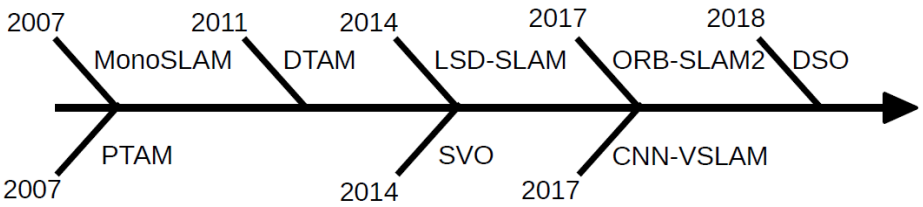


Fig. 2.8 Timeline representing the most important visual-only SLAM algorithms (adapted from [63]).

ORB-SLAM2 The most advanced feature-based algorithm is the ORB-SLAM2 algorithm [72], which is built on the ORB-SLAM algorithm [76]. It runs tracking, local mapping,

and loop closure in three separate processes. By identifying feature correspondences and reducing the reprojection error, the first thread locates the sensor. Map maintenance tasks are carried out by the local mapping process. The final component, known as loop closure, is in charge of finding new loops and fixing drift errors within existing loops. The algorithm then completes a comprehensive bundle adjustment after handling the three threads, taking into account the entire structure and approximated motion consistency. The components that make up the program are illustrated in Figure 2.9. The ORB-SLAM2 algorithm uses global optimization and loop closing methods while taking into account the monocular, stereo, and RGB-D approaches. However, if the system fails to identify a high-similarity frame, the monitoring failure scenario could result in a lost state [77]. Real-time operation in embedded platforms is challenging because this technique must gather the images at the same frame rate as it analyzes them [78]. Despite the reality that there are a number of embedded versions documented in the literature, this is the case. Both in [79] and in [78] ORB-SLAM algorithm has been applied on various CPU- and GPU-based systems, and the authors assessed the performance of each thread on each platform. An enhanced version of the ORB-SLAM2 algorithm has been developed as part of this thesis. Specifically, the ORB-SLAM2 algorithm has been heavily modified in order to increase the number of parameters controlling the algorithm, to include compilation procedures for ARM processors, and to integrate several optimizations for Nvidia CUDA GPUs. The software is provided open source in [80]. The ARM adaptation forced us to disable all the optimizations related to the x86 architecture, with a consequent performance drop especially in the feature extraction and computation phases, where AVX/SSE vector instruction sets have been widely used. For that reason, the optimization for CUDA GPUs has been taken into account as a mandatory activity in order to address the performance issues.

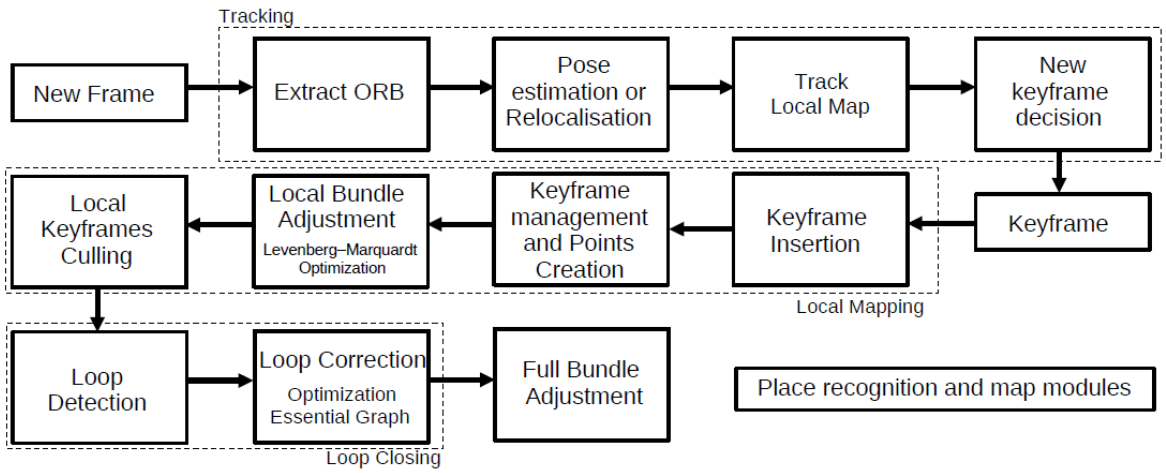


Fig. 2.9 Diagram depicting the ORB-SLAM2 algorithm (adapted from [72]).

Visual-SLAM Deep Learning-Based Algorithms

Recent works propose the incorporation of deep learning techniques [81] to increase the system's robustness in the Visual-SLAM system. The UnDeepVO [82] is a noteworthy program that applies deep learning principles. Through a deep neural network, this monocular visual-odometry method can estimate posture and depth. The authors use stereo pictures to train UnDeepVO through unsupervised learning, and they also take both spatial and temporal rich information into account when calculating the training's loss function. In comparison to other monocular techniques like the ORB-SLAM, this approach was found to be more reliable and precise (without loop closure). The DeepSLAM was recently suggested by the same study team [83]. The system takes into account tracking- and mapping-nets that were learned using unsupervised learning and took into account spatial and temporal geometry in the loss function. A Loop-Net is also included in the method to carry out loop identification. In comparison to other binocular algorithms like ORB-SLAM, DeepSLAM demonstrated superior performance as well as greater stability. The DF-SLAM [84] is another pertinent deep learning method. In contrast to ORB-SLAM, which employs hand-made ORB features, DF-SLAM utilizes deep local features that are described by the TFeat network. The authors compare DF-SLAM and ORB-SLAM2 in a number of findings; for the majority of sequences, the suggested method performed better. Recently, several overviews ([85], [86], [87]) that cover deep learning-based methods used for depth estimation and the fundamentals of SLAM's orientation have been published in the literature.

2.1.6 Multi-sensor fusion for robot perception

The Joint Directors of Laboratories Data Fusion Working Group came up with the idea for the data fusion process paradigm. This model serves as a general blueprint for a data fusion system and was developed to serve as a foundation for the development of numerous data fusion methods [21]. The connections between the data sources and the information extraction procedures are specified by the fusion model. Different processing levels can be found between the data extraction and the end information provided to judgment stages, according to the authors of [21]:

- Source processing: creates preliminary information from raw data.
- Object refinement: refines the preliminary information to identify objects.
- Situation refinement: establishes the relations among identified objects.
- Threat refinement: tries to infer details about future states of the system.

- Process refinement: analyses the performance of the previous levels and determines if they can be optimized.
- Data management: takes care of the storage management of the processed data.

The process of combining multiple views from various sensor sources is known as multi-sensor data fusion, and it aims to produce a more accurate, robust, and comprehensive depiction of the environment of interest. It is anticipated that the fused representation will outperform the sources' separate outputs. We will cover the most recent multi-sensor fusion research from two angles in the parts that follow. The most popular techniques for fusing sensing data will be examined first, with an emphasis on their benefits and shortcomings. Second, we'll look at the various layers of the perception issue where fusing can be done. The final viewpoint enables us to examine the benefits of carrying out fusion at each stage and to concentrate on the problems that still need to be resolved in the associated fusion approaches.

Fusion methodologies

The majority of multi-sensor fusion approaches are based on probabilistic techniques, but techniques based on the theory of evidence have been suggested as an option to many categories of robot sensing in addition to multi-sensor fusion. We will go over the primary fusing strategies within these two options in the subsections that follow. After that, we'll go over the various multi-sensor fusion designs for intelligent robot awareness.

Probabilistic methodologies The probabilistic methodologies rely on the Bayes' rule, which provides means to make inferences about an event described by a state x, given a measurement observation z. The relationship between x and z is encoded in the joint probability distribution $P(x,z)$ which can be expressed as follows:

$$P(x,z) = P(x|z)P(z) = P(z|x)P(x), \tag{2.17}$$

then, the Bayes' rule in terms of the conditional probability $P(x|z)$ is:

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)}. \tag{2.18}$$

In equation (2.18), $P(x|z)$ represents the probability of event x occurring given that event z has occurred. $P(z|x)$ represents the probability of event z occurring given that event x has occurred. $P(x)$ and $P(z)$ represent the probabilities of events x and z, respectively (the prior

probabilities). To obtain more information about the state x, an observation z is made. These observations are modelled in the form of the conditional probability $P(z|x)$ which describes, for each fixed state x, the probability that the observation z might be made. New likelihoods associated with the state x are computed from the product of the original prior information ($P(x)$) and the information gained by an observation ($P(z|x)$). This information is embedded in the posterior probability $P(x|z)$ which describes the likelihoods associated with x given the observation z. In this fusion process, the marginal probability $P(z)$ is used to normalize the posterior probability. $P(z)$ plays an important role in model validation or data association as it provides a measure of how well the observation is predicted by the prior. Equation (2.18) provides a principled means of combining observed information with prior beliefs about the state of the world.

Conditional probability $P(z|x)$ functions as a sensing model in driving uses. The probability $P(z|x)$ is created when creating a sensor model by first putting the value of $x = x'$ and then computing $P(z|x = x')$. Alternatively, $z = z'$ is fixed and a probability function $P(z = z'|x)$ is derived when this sensor model is applied and observations are made. The proportional probability that various values of x result in the observed value of z is represented by the likelihood function. The posterior, or observation update $P(x|z)$, is created by multiplying this probability by the prior, which is also specified on x. The Bayes' rule can be used for multi-sensor fusion and it requires conditional independence that can be expressed as follows:

$$P(x|Z^n) = \mathrm{C}P(x)\prod_{i=1}^{n} P(z_i|x), \tag{2.19}$$

where C is a normalizing constant. The equation (2.19) states that the posterior probability on x given all the observations Z^n is proportional to the product of prior probability and individual likelihoods from each source of data (sensor). So, the Bayes' rule can be written in the recursive form as follows:

$$P(x|Z^k) = \frac{P(z_k|x)P(x|Z^{k-1})}{P(z_k|Z^{k-1})}, \tag{2.20}$$

where $P(x|Z^k)$ contains all the past information, thus, when a new information $P(z_k|x)$ arrives, the previous posterior takes on the role of the current prior and the product becomes the new posterior, after normalization.

Occupancy grids Conceptually, using probabilistic occupancy grids (POGs) is the easiest way to apply Bayesian data merging techniques. POGs, despite being straightforward, can be used to solve a variety of issues related to the perception job, such as mapping ([39]

and [19]), moving object recognition ([88] and sensor fusion ([89]). When mapping, the area of interest is discretized into a matrix of spatial units of uniform size. Each cell contains a probability number, $P(x_{i,j})$, that indicates the occupancy status of the cell; typically, a cell can be either vacant or filled. Maintaining a probability distribution on potential state values $P(x_{i,j})$ at each grid cell is the objective. Bayesian techniques require sensor models or likelihood functions to fill the matrix after the state has been established. In order to do this, it is necessary to define the probability distribution $P(z|x_{i,j})$ that links each potential grid state to a distribution of data. This is typically done as an observation grid per sensor input, resulting in a grid of likelihoods over each $x_{i,j}$ occupied state for a given observation $z = z'$. One can see that the precision and scale of the surroundings have a significant impact on the processing cost of keeping an updated fused occupancy grid. Grid-based fusing is suitable when the domain size and scale are minimal or when it is possible to make inferences about the environment that will result in a smaller grid and lower updating costs. Grid-based techniques offer simple and efficient merging strategies in these situations. Grid-based techniques can be enhanced in a variety of ways, including by using hierarchical (quadtree) grids ([90]), unusual (triangular, pentagonal) grids ([91]), or working with local map assumptions to prevent a process of global map updating.

Kalman filter The Kalman filter (KF) is a recursive linear predictor that, using periodic measurements Z of the state, iteratively determines an estimate for a continuously evolving state x. KF uses an explicit statistical model to describe the evolution over time of the parameter of interest $x(t)$ and an explicit statistical model to describe the relationship between the observations $z(t)$ and this parameter. To depict the conditional mean $\hat{x}(t) = E[x(t)|Z^t]$, the gains used in a KF are selected to guarantee that the resulting approximation $\hat{x}(t)$ minimizes the mean-squared error. Due to its characteristics, KF is well adapted to handle multi-sensor estimation and data integration issues. First, the clear description of the processes and observations makes it possible to integrate a broad range of various sensor models into the fundamental algorithm. Second, it is feasible to numerically assess each sensor's contribution to system performance through the consistent application of statistical measures of uncertainty. The algorithm's linear iterative structure also guarantees that its implementation is straightforward and effective. The random variables defining process and measurement noise are all assumed to be Gaussian, temporally uncorrelated, and zero-mean as a fundamental presumption in the development of the Kalman filter. The Kalman filter's findings could be deceptive if these restrictions are not met. In these circumstances, more advanced Bayesian filters are typically used. When at least one of the state model or the data model is nonlinear, the extended Kalman filter (EKF), a modification of the Kalman filter, may be used ([92]).

The non-linear variant of KF is known as EKF. The unscented Kalman filter is another non-linear variant of KF (UKF). In the UKF, the underlying Gaussian distribution is represented by a fixed selection of points that approximates the probability density. These values will be transformed non-linearly in order to estimate the posterior distribution. The process is referred to as the unscented transform. In terms of error prediction, UKF is typically more reliable and precise than the EKF when strong non-linearities are there ([93]).

Monte Carlo methods Probability distributions are described by Monte Carlo (MC) techniques as a collection of weighted observations from an underlying state space. These examples are then used by MC filtering to mimic Bayes' rule-based probabilistic reasoning. Numerous models or examples are run. A probabilistic behavior of the replicated process is found by examining the statistics of these samples as they move through the inference process ([94]). When state transition models and data models are extremely non-linear, MC techniques work well. Because sample-based techniques can depict very broad probability densities, this is the case. Particularly, Monte Carlo methods are good at handling multi-modal or numerous hypothesis density functions ([39]). MC techniques are thought to fall somewhere between grid-based and parametric data merging techniques. However, in situations where the state space has a large dimension, MC techniques might not be suitable. This is due to the fact that the state space dimension grows exponentially with the number of samples needed to acquire a correct model. Fortunately, by marginalizing out states that can be described without sampling, the dimensionality increase can be restrained.

Limitations of probabilistic methods The issue of information representation, and consequently information integration, is significantly impacted by uncertainty representation. The representation of random uncertainty is well adapted for probabilistic techniques, but they do not advocate a clear picture of imprecision. We can enumerate the major problems with probabilistic techniques for information fusion in terms of their perceptual constraints:

- Complexity: the need to specify a large number of probabilities to be able to apply probabilistic reasoning methods correctly.
- Inconsistency: the difficulties involved in specifying a consistent set of beliefs in terms of probability and using these to obtain consistent deductions about the events or states of interest.
- Precision of models: the need to have precise specifications of probabilities about barely known events.

- Uncertainty: the difficulty in assigning probability in the face of uncertainty, or ignorance about the source of information.

Furthermore, three main theories that intend to overcome the previous limitations of probabilistic methods should be mentioned: interval calculus, fuzzy logic and theory of evidence.

Interval calculus Based on the concept of expressing uncertainty by using an interval to limit actual parameter values, interval calculus (IC) was developed ([95]). Using IC could have benefits over random approaches. In cases where there is a dearth of probabilistic knowledge but where sensor and parameter error is known to be bounded, intervals offer a reasonable measure of uncertainty. The actual value of the state x is known to be limited from below by a and from above by b, where $x \in [a,b]$, according to IC methods. This assertion serves to simply characterize the ambiguity in the parameter x. In particular, the assertion $x \in [a,b]$ does not necessarily suggest that x is equally likely (uniformly distributed) over the range $[a,b]$. It is crucial that no other extra probabilistic structure is inferred. Object recognition techniques based on IC are sometimes employed. The inability to obtain results that converge to a desired value and the difficulty of encoding dependencies between variables, which are at the heart of many data fusion issues, such as variables defining the state and appearance of a moving object, prevent them from being widely used in data fusion.

Fuzzy logic A well-known technique for expressing uncertainty in management and data fusion uses is fuzzy logic. It works with approximate thinking as opposed to precise reasoning. According to [96], fuzzy reasoning is founded on degrees of truth rather than absolute numbers. Probabilities and degrees of truth are essentially separate; degrees of truth reflect membership in ill-defined sets, not the probability of an occurrence or circumstance. Fuzzy logic offers a rigorous mathematical structure that enables the exact and rigorous representation and study of ill-defined conceptual events. It is also suitable for circumstances where fuzzy connections, conditions, and events occur as a modeling language. In contrast to fuzzy sets, where membership is based on a degree between the potential absolute values, traditional logic sets have binary membership, where a variable is either in the set or it is not. Truth values, vocabulary (operators), and reasoning processes can be used to identify logic as the foundation for reasoning (tautologies, syllogisms). Dual logic uses boolean truth tables to describe its operators and allows truth values to be either true or false. The truth values in fuzzy logic are represented by language variables true and false rather than being limited to the two values true and false. Although fuzzy logic is frequently used in control applications, its constraints are taken into consideration in sensor integration for driving applications. It can only a small degree of participation duties. In addition, choosing

appropriate membership functions and fuzzy rules is challenging and gets more difficult as the number of sensing inputs or items of interest rises. Additionally, thorough testing is necessary for the fuzzy system's proof and confirmation, which is crucial in systems where safety is a key consideration [97].

Evidence theory A significant option to probability theory is evidence theory (ET), also known as the Dempster-Shafer theory of evidence [98]. It has demonstrated particularly significant success in uses requiring automatic logic, such as intelligent driving systems. Evidential thinking differs fundamentally from fuzzy set theory and probabilistic techniques. Let's think about a global collection Ω of all potential hypotheses for an occurrence x. A belief mass (likelihood of a proposition) may be assigned to any member of $a \in \Omega$ and, in fact, to any subset $A \subseteq \Omega$ in probability theory or fuzzy set theory. Belief mass can be assigned to sets of sets as well as components and sets in evidentiary reasoning. The power set 2^{Ω} is the domain of evidential reasoning, whereas the domain of probabilistic techniques is all potential subgroups Ω. Dempster-Shafer's approach, in a nutshell, seeks to measure levels of belief. There are various types of inaccurate information, such as unclear or vague information, in the area of intelligent car awareness. For instance, when an object is absent (occlusions), when a sensor cannot measure all of the object's pertinent characteristics (hardware constraints), or when an observation is unclear (partial object detection). Uncertainty and ambiguity both contribute to agents' views and subjective judgments about the actual value of a relevant variable [99].

Transferable Belief Model The Dempster-Shafer theory is interpreted by the Transferable Belief Model (TBM), which is built on the work in [100]. According to Shafer, this model is a pure version of Dempster-Shafer's model that does not include any references to probability theory. Let the collection of all potential answers to an issue, where each of its components is mutually exclusive, be the frame of discernment (or space of hypotheses) Ω. A confidence function with the range $[0,1]$ as the image and the power set 2^{Ω} as the domain can be used to quantify the information of the world that the agent $\mathscr{Y}$ possesses. The TBM has the ability to expressly depict doubt regarding a Ω hypothesis. It reflects the belief for the known hypotheses and takes into consideration what is still unknown. The real answer to the issue corresponds to one of the elements of Ω, denoted ω, but the agent is unsure of which element because of its imprecision. Regarding the possibility that a particular subgroup of Ω contains ω, the agent can only offer his subjective view. Based on the information that was accessible to $\mathscr{Y}$ at the time t, the belief $bel(A)$ provided by $\mathscr{Y}$ to a subset A of Ω at that time expresses how strongly the agent believes that ω is an element of A. A likelihood indicator is

typically used to quantify the level of confidence. The Transferable Belief Model concerns the same problem as the one considered by the Bayesian model except it does not rely on probabilistic quantification but on belief functions [101].

The TBM is a two-level model. At the *credal* level, where the individual states the degree to which he believes that ω pertains to particular subgroups of Ω, beliefs are defined by belief functions. The information is combined, updated, and kept at this stage. When a choice needs to be made, the *pignistic* level emerges, and the views held at the *credal* level pass on the knowledge required for the *pignistic* level to make the best choices. When there are no decisions to be made, this stage is idle. TBM posits that belief functions can quantify ideas held at the *credal* level. The TBM postulates that the impact of a piece of evidence on an agent is transformed by an allocation of parts of an initial unitary amount of belief among the subsets of Ω. For $A \subseteq \Omega$, $m(A)$ is a part of the agent's belief that supports A. The $m(A)$ values, $A \in \Omega$, are called the basic belief masses (*bbm*) and the m function (or mass function) is called the basic belief assignment (*BBA*) and is defined as:

$$\begin{aligned} m(A) &\rightarrow [0,1], \quad \text{with:} \\ \sum_{A \subseteq \Omega} m(A) &= 1, \quad m(\oslash) = 0. \end{aligned} \tag{2.21}$$

Every $A \in \Omega$, such that $m(A) > 0$, is called a focal element (or proposition). The difference with probability models is that masses can be given to any subsets of Ω instead of only to the elements of Ω as it would be the case in probability theory. From Equation (2.21) it can be noticed that a *BBA* may support a set A without supporting any of its subsets. This can be seen as a partial knowledge capability. A can be any subset in 2^{Ω}, but there are some cases where the mass function $m(A)$ has a specific meaning:

- If there is only one focal set, $m(A) = 1$ for some $A \subseteq \Omega$, then $m(A)$ becomes a categorical mass function. And if $A = \Omega$ this represents the total ignorance.
- If all focal sets are singletons, $m(A) > 0 \rightarrow |A| = 1$, $m(A)$ could be considered a Bayesian mass function.

The reason for it is called transferable belief model is the following: if there is new information about $\omega \in B \subseteq \Omega$. Then, this results in an evidence transfer for each $A \subseteq \Omega$ of the *bbm* $m(A)$ initially allocated in A, to $A \cap B \subseteq \Omega$.

In the TBM, evidence reliability is a key idea. When an unreliable source provides the evidence masses given in the *BBA*, those masses should be regarded as being inaccurate. By weighing the bulk assignations, a discounting factor enables the introduction of this absence of dependability. The original transfer of belief described in the TBM corresponds to the

unnormalized rule of conditioning (or Dempster's rule of conditioning). Let us assume that conditioning evidence tells the agent $\mathscr{Y}$ that $B \subseteq \Omega$ is true. Then, we can transfer the original *BBA* m into an updated *BBA* m_B as per the following formula:

$$\begin{aligned} m_B(A) = \sum_{X \subseteq B} m(A \cup X), \quad A \subseteq B, \\ m_B(A) = 0, \quad A \nsubseteq B, \end{aligned} \tag{2.22}$$

where m is a *BBA* on the frame of discernment Ω.

The degree of belief $bel(A)$ of $A \subseteq \Omega$ quantifies the total amount of justified support given to A. This can be obtained summing all the basic belief masses given to propositions $X \subseteq A$, with $X \neq \oslash$:

$$\begin{aligned} bel: \quad & 2^{\Omega} \rightarrow [0,1], \\ bel(A) = \quad & \sum_{\oslash \neq X \subseteq A} m(X), \end{aligned} \tag{2.23}$$

which is evident as $bel(A)$ includes evidence only given to specific subsets of A. The function bel is called a belief function and satisfies the following inequalities as proposed in [100]:

$$\begin{aligned} A_1, A_2, \ldots, A_n \quad & \subseteq \quad \Omega \\ bel(A_1 \cup A_2 \cup \ldots A_n) \quad & \geq \quad \sum_i bel(A_i) - \sum_{i>j} bel(A_i \cap A_j) \cdots - (-1)^n bel(A_1 \cap A_2 \cap \ldots A_n), \\ \forall n \quad & \geq \quad 1. \end{aligned} \tag{2.24}$$

A big lack in probability models is the ability to represent total ignorance as state of belief. This state represents a problem in Bayesian theory when it has to be defined by probability functions. In the TBM, total ignorance is represented by a vacuous belief function $m(\Omega) = 1$, so that $bel(A) = 0, \forall A \subseteq \Omega, A \neq \Omega$, and $bel(\Omega) = 1$. All the subsets $A \in \Omega$ receive the same degree of belief (which is the state of total ignorance) and none of the are supported (except for Ω itself). The degree of plausibility $pl(A)$ of $A \subseteq \Omega$ quantifies the maximum amount of potential support that could be given to $A \subseteq \Omega$ and it is obtained by adding all the basic belief masses given to propositions X that are compatible with A (i.e. $X \cap A \neq \oslash$):

$$\begin{aligned} pl: \quad & 2^{\Omega} \rightarrow [0,1], \\ pl(A) = \quad & \sum_{X \cap A \neq \oslash} m(X) = bel(\Omega) - bel(\bar{A}), \end{aligned} \tag{2.25}$$

meaning that there are basic belief masses included in $pl(A)$ that could be transferred to non-empty subsets of A if some new information could justify that transfer, for instance, if we know that $\bar{A}$ is impossible. The function pl is called plausibility function and it is an alternative representation of the information represented by the belief function over the same *BBA*:

$$pl(A) = bel(\Omega) - bel(\bar{A}) = \sum_{A \cap B \neq \oslash} m(B). \tag{2.26}$$

Fusion of evidence An important benefit of the evidential theory, which can be thought of as an information fusing operation, is its capacity to combine multiple pieces of evidence from various sources within the same framework of judgment. We can define two evidence distributions m_1 and m_2 with elements from two different sources S_1 and S_2, respectively. These two evidence bodies with focal elements $X_1, X_2, \ldots, X_i$ and $Y_1, Y_2, \ldots, Y_j$ can be combined into a new mass function m using the combination rule. Specifically, the product of m_1 and m_2 induced by the two bodies of evidence on the same frame of discernment Ω supports $X \cap Y$. This rule provides a method to compute the orthogonal sum $m = m_1 \oplus m_2$ as follows:

$$\begin{aligned} m(A) &= \frac{\sum_{X_i \cap Y_j = A} m_1(X_i) m_2(Y_j)}{1 - K} \quad \text{for} \quad A \subset \Omega, \\ K &= \sum_{X_i \cap Y_j = \oslash} m_1(X_i) m_2(Y_i) \\ m(\oslash) &= 0, \end{aligned} \tag{2.27}$$

where K is the conflict factor (formerly a normalization factor) that measures the degree of conflict evidence between the bodies of evidence m_1 and m_2. If the value of the conflict factor is high, the conflict is strong between the sources; therefore, a combination would make no sense.

The normalization factor in Equation (2.27) produces convergence toward the dominant opinion between the bodies of evidence to be combined. Specifically, concordant items of evidence (redundant evidence) reinforce each other by transferring mass in the null set $\oslash$ to the focal elements. However, when the bodies of evidence are not reliable or the mass functions are imprecise, a conflict mass $m(\oslash)$ appears. Particularly, when there is a considerable degree of conflict between the sources of evidence, the normalization process inside (2.27) of combination can lead to counterintuitive results. For example, it can lead to assign a high belief to a minority element when no agree is achieved between the evidence sources. In [102] an alternative to Equation (2.27) to avoid counter-intuitive results when a high conflict value is present in the evidence combination has been proposed. This alternative

is a rule of combination that assigns the conflict mass from the null set $\oslash$ to the ignorance set Ω, which means that the conflict value is distributed among all the elements of the frame of discernment rather than only the elements with intersections of the combining masses. The rule of combination proposed in [102] is defined as follows:

$$\begin{aligned} m(A) &= \sum_{X_i \cap Y_j = A} m_1(X_i) m_2(Y_j) \quad A \neq \oslash, A \neq \Omega, \\ m(\Omega) &= \sum_{X_i \cap Y_j = \Omega} m_1(X_i) m_2(Y_j) + K \\ K &= \sum_{X_i \cap Y_j = \oslash} m_1(X_i) m_2(Y_i). \end{aligned} \tag{2.28}$$

Analysis of the evidence theory for robot perception The ability of ET to depict insufficient proof, full ignorance, and the absence of a need for *a priori* odds is its first benefit. The complete belief is attributed to ignorance before the process of gathering proof. So when new proof is discovered, it takes the position of the old evidence in the ignorance. Although implicit knowledge is embedded in the description of the framework of the frame of discernment, ET does not necessitate *a priori* environmental information. Discounting variables, such as sensor performance dependability, are a key method for incorporating the credibility of the proof sources into the ET representation. Sensor fusion apps can take advantage of the capacity of combination rules to combine data from various bodies of evidence into a more trustworthy evidence distribution. The various ways that belief functions are interpreted make it possible to choose the best decision-making instrument for the intended purpose. Using the appropriate evidence representation, advanced phases of intelligent robot systems, such as thinking and judgment, can incorporate evidence distributions into the decision-making process. The representation and the synthesis of the data are the two major drawbacks of ET. Due to the belief functions' distribution of belief to the 2^{Ω} power set of all the hypotheses, ET is firstly less computationally tractable as the number of hypotheses rises. However, the application area might permit making assumptions to limit the collection of potential hypotheses by Ω. By using the right meaning of the frame of discernment, this drawback might be surmounted. Second, the normalization process that emerges from the divergent values among the masses of data produces unexpected outcomes. However, ideas like the rule expressed in (2.28) circumvent this restriction by handling the dispute rather than averaging the collected data.

Deep learning approaches There are several deep learning techniques that can be applied to robot multi-sensor fusion, based on artificial neural networks (ANN) and deep neural network (DNN) architectures (see Figure 2.11 and Figure 2.10 for reference).

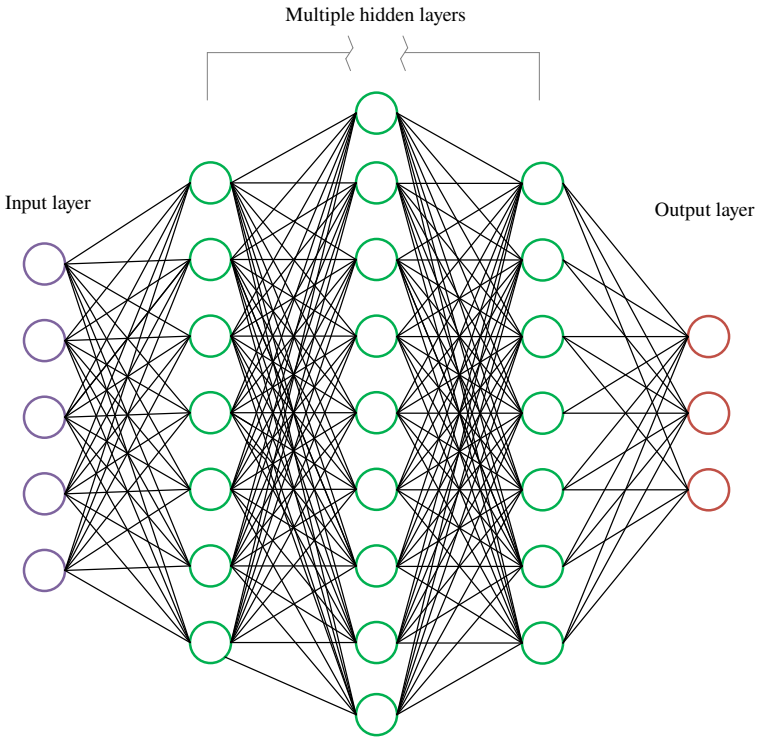


Fig. 2.10 Architecture of a Deep Neural Network.

Convolutional Neural Networks (CNNs) that are often used for visual perception tasks, such as object recognition and detection, can be applied to data from cameras or other visual sensors to perform multi-sensor fusion. The CNNs have been used to perform sensor fusion in several applications: the authors in [103] proposed a solution for predicting occupancy using multiple low-cost and low-resolution heat sensors, performing the data fusion and processing via a Convolutional Neural Network for predicting occupancy. In [104], the authors proposed a multi-sensor fusion approach based upon convolutional neural network to integrate layer-wise images, acoustic emission signals, and photodiode signals for in-situ quality monitoring of Selective Laser Melting (SLM). Specifically, the authors designed three CNN-based multi-sensor fusion models from data-level fusion, feature-level fusion, and decision-level fusion respectively and compared in quality identification.

Another approach to sensor fusion is using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. RNNs are useful for processing sequential data, such as sensor readings over time. They can be applied to data from sensors that provide time-series data, such as accelerometers, gyroscopes, or temperature sensors. LSTM

networks are a type of RNN that can handle longer sequences of data and are often used for time-series data. They can be applied to sensor data that changes over time, such as positional data or environmental data. The authors in [105] proposed a combined graph convolutional network (GCN) and long short-term memory (LSTM) algorithm to construct a feature-level fusion model for sensor data. In the paper, the authors proposed a model that uses GCN to extract features of multi-source heterogeneous data, which solving the problem of difficult fusion of heterogeneous data caused by differences in data types, and LSTM for feature extraction of time series, which solves the problem of gradient disappearance.

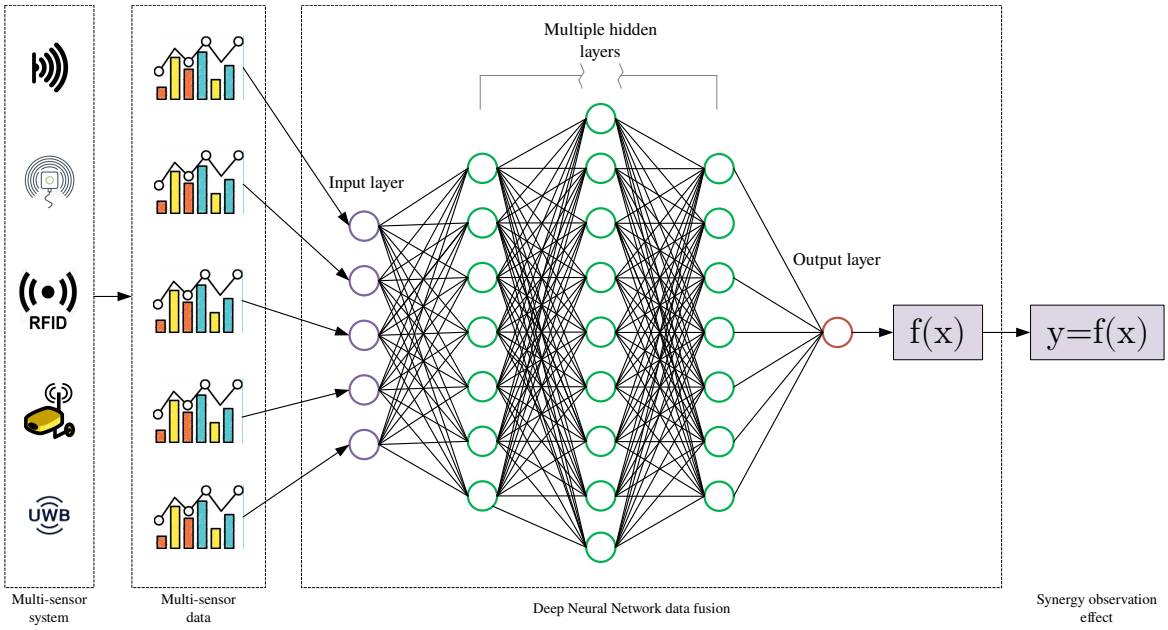


Fig. 2.11 Diagram depicting a framework for multi-sensor fusion based on deep neural networks.

A further approach to sensor fusion with DNN is using autoencoders. Autoencoders are used for unsupervised learning and can be used to learn representations of multi-modal data. They can be applied to sensor data from different modalities, such as visual, auditory, and tactile sensors. In [106], the authors proposed a multi-sensor fusion framework using a novel autoencoder for semi-supervised learning. Specifically, both labeled and unlabeled data are used for learning the latent representation from each sensor. Then, the latent representation of all the sensors are combined to perform classification. A joint optimization formulation is presented for learning the sensor-specific latent representation, their encoder and decoder weights and the classification weights together. In [107], the authors presented a multi-sensor data collection and data fusion procedure for nondestructive evaluation/testing (NDE) of a concrete bridge deck. The authors trained a neural network autoencoder to quantify the relationship between several NDE results using the data collected at common positions. This

relationship was then used by the authors for fusion of NDE data to increase the reliability and spatial resolution of the NDE measurements and to generate a data-fused condition map showing novel characteristics.

Generative Adversarial Networks (GANs) can be used to generate synthetic data, which can be used to augment existing sensor data. This can help to improve the robustness of the system and make it more resistant to sensor failures or environmental changes. Even if GANs are used especially to generate synthetic data, these networks can also be used to solve multi-sensor fusion problem. In [108] the authors proposed a data driven approach to multi-modal fusion, exploiting selected optimal features from an estimated latent space of data across all modalities. The authors used a generative network conditioned on individual sensor modalities to learn the hidden space. The hidden space, as an intrinsic structure, has been exploited in detecting damaged sensors, and in subsequently safeguarding the performance of the fused sensor system.

These deep learning techniques can be used to fuse data from multiple sensors to create a more accurate and comprehensive representation of the robot's environment, enabling the robot to make more informed decisions and navigate effectively.

Requirements of multi-sensor systems

We discussed the issue with single-sensor perception systems in the earlier parts, as well as the possibility of incorporating multiple sensors into the perception job. We need to suggest and put into practice a number of components in order to create a multi-sensor system. First, sensor data processing gets the raw data from instruments and turns it into usable information by using sensor-based models. The quantity and type of sensors placed on the demonstrator determine how raw data processing units are configured. After that, SLAM modules use the data that has been analyzed to make a plan and carry out localization. At this point, these components might need to perform data fusion.

The most crucial specifications for a multi-sensor system for independent robots are summarized here:

- Sensor Diversity: The system should have sensors that can sense different types of data, such as visual, auditory, or tactile. This diversity helps to provide a more complete understanding of the robot's environment.

- Redundancy: The system should have multiple sensors that can sense the same type of data, such as multiple cameras or multiple microphones. This redundancy helps to ensure that the robot can still function even if one or more sensors fail.

- Accuracy: The sensors should be accurate enough to provide precise data that the robot can use to navigate and make decisions.
- Reliability: The sensors should be reliable enough to provide consistent data over time, even in challenging environments.
- Low Latency: The sensors should be able to provide data quickly enough for the robot to make real-time decisions.
- Integration: The sensors should be integrated with the robot's control system to enable the robot to use the data to make decisions and take actions.
- Power Efficiency: The sensors should be designed to consume as little power as possible, to prolong the robot's battery life.
- Scalability: The system should be scalable, so that additional sensors can be added or removed as needed, depending on the robot's specific task and environment.

Overall, a multi-sensor system for autonomous robots should be designed to provide a robust, reliable, and accurate perception of the robot's environment, enabling the robot to make informed decisions and navigate effectively.

2.2 Resilience in robot perception

Robots have become an integral part of our daily lives, from manufacturing and assembly lines to homes and healthcare. As robots become more ubiquitous, their perception of the environment and ability to adapt to changing conditions become critical factors for their success. Resilient robot perception is the ability of a robot to accurately perceive and interpret its environment facing uncertainty, noise, and adversarial conditions. This requires advanced sensing technologies, robust algorithms, and intelligent decision-making systems that can adapt to changing environmental conditions. In this section, we will explore the challenges of robot perception and the strategies and techniques for building resilient perception systems that can operate in real-world scenarios. We will discuss the latest research and advancements in the field, presenting robot perception architectures, multi-modal perception and the outlier detection on the measurements to achieve resilience. Finally, we will examine the potential applications of resilient robot perception in various domains, including autonomous robots.

2.2.1 Robot perception architecture

Many contemporary multi-sensor fusion techniques are based on the fusion ideas discussed in earlier sections. These ideas are being put into practice at various phases of the robot perception system. In order to explain the various state-of-the-art techniques linked to our efforts from a different angle, we adhere to the basic design for a robot perception architecture suggested in [42]. The two fusion stages that we take into account within a perceptual system are shown in Figure 2.12:

- Low level. Each sensor's raw data is converted into a shared representation, which is then fused into a representation that is used to create the image.
- Map level. The produced maps are merged to create a fused map after SLAM is solved for each sensor output.

Within the SLAM component, low level and map level fusions are carried out. As previously mentioned, perception is predicated on the notion that numerous sensors can provide duplicate information (i.e., overlapping field of views) as well as complimentary information (for example, LiDAR, UWB, UHF-RFID, and camera sensor). The processes of sensor setup and selection are closely linked to the use of intelligent robots. As a result, the suggested multi-sensor fusion techniques from the state-of-the-art have a wide range of sensor setups. We concentrated on techniques that use camera and range sensors in order to target the connected works. This choice is made based on the sensor configuration we employ to evaluate our suggested multi-sensor fusing methods.

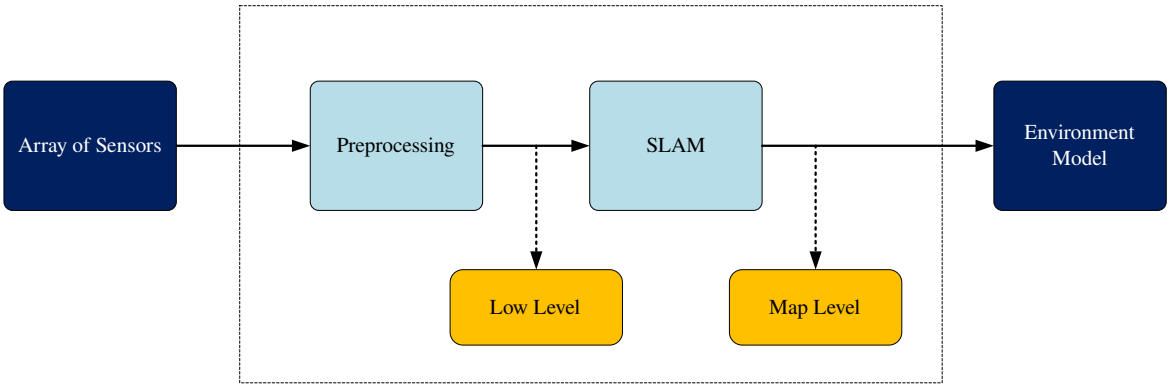


Fig. 2.12 Fusion levels for robot perception architecture.

2.2.2 Multi-modal perception

The robot ability to perceive the world around them is still limited. Single-mode sensors like cameras or LIDAR are useful, but they have limitations. Multi-modal perception is

an approach that combines data from multiple sensors and information sources to improve the accuracy and reliability of robot perception. This section will explore the benefits and challenges of multi-modal perception, as well as techniques for integrating data from different sources.

Multi-modal perception has several advantages over single-mode sensors. By using multiple sensors, robots can capture more comprehensive information about their environment, including visual, auditory, and tactile cues. This additional information can help robots make better decisions, especially in situations where a single sensor may be insufficient, such as low-light conditions, noisy environments, or occlusions. Additionally, multi-modal perception can help robots interpret the environment more accurately by combining data from different sources and providing redundancy in case one sensor fails.

While multi-modal perception has significant benefits, it also presents several challenges. One of the most significant challenges is how to integrate data from different sources effectively. The data from different sensors may be in different formats or have different levels of uncertainty, making it challenging to combine them effectively. Additionally, the data from different sensors may arrive at different times, which can complicate the integration process further.

There are several techniques for multi-modal perception, including sensor fusion, machine learning, and Bayesian inference. Sensor fusion is a process of combining data from multiple sensors to obtain a more comprehensive understanding of the environment as presented in Section 2.1.6. Machine learning approaches can be used to learn how to combine data from different sources effectively, especially when the data from different sensors are in different formats as reported in Section 2.1.6. Finally, Bayesian inference is a statistical approach that can be used to integrate uncertain data from multiple sources.

Multi-modal perception has several potential applications, including autonomous robots, healthcare, and manufacturing. In autonomous robots, multi-modal perception can help robots navigate safely in challenging conditions, such as low-light or inclement weather. In healthcare, robots with multi-modal perception can assist in surgery or rehabilitation, providing more accurate and precise feedback to surgeons or patients. In manufacturing, multi-modal perception can help robots perform more complex tasks, such as assembling products or inspecting parts.

Multi-modal perception is a powerful approach to improve the accuracy and reliability of robot perception. By combining data from multiple sensors and information sources, robots can better understand and interpret their environment, even in challenging conditions. While multi-modal perception presents several challenges, there are several techniques and approaches that can be used to overcome them. In [109], the authors developed a graphical

model for fusing object recognition results using two different modalities–computer vision and verbal descriptions. In this work, the authors focused on three types of verbal descriptions, egocentric positions, relative positions using a landmark, and numeric constraints. Visual and verbal modalities have been fused using a Conditional Random Fields (CRF) based approach, modelling n-ary relations (or descriptions) as factor functions. Furthermore, in the context of service robotics, the authors in [110] proposed a multi-modal perception based framework to realize non-intrusive domestic assistive robotic system. All the robot's actions are based on multi-modal perceptions which include user detection based on RGB-D data, user's intention-for-interaction detection with RGB-D and audio data, and communication via user distance mediated speech recognition. Finally, the authors in [111] presented an active fusion framework for the multi-modal material recognition. The authors adopted an adversarial learning method to obtain the modal-invariant representations to bridge the gap between different modalities, and then they developed a reinforcement learning method for active modality selection. The developed framework and algorithms showed promising material recognition results.

2.2.3 Outlier detection

Outlier detection in sensor measurements is a critical aspect of achieving resilience in robotic systems [112]. In the context of robotic systems, outliers in sensor measurements can arise due to a variety of reasons, such as noise, environmental conditions, or sensor malfunctions. Outlier detection is the process of identifying and removing or correcting these outlier data points to ensure accurate and reliable measurements. An outlier is a data point that is significantly different from other data points in the same set. Figure 2.13 shows outliers in a simple 2-D data set. The data has two normal regions, N_1 and N_2. O_1 and O_2 are two outlying instances while O_3 is an outlying region. The outlier instances are the ones that do not lie within the normal regions. Outliers exist in almost every real dataset. Some of the most important causes for outliers are:

- Malicious activity - such as insurance or credit card or telecommunication fraud, a cyber intrusion, a terrorist activity
- Instrumentation error - such as defects in components of machines or wear and tear
- Change in the environment - such as a climate change, a new buying pattern among consumers, mutation in genes
- Human error - such as an automobile accident or a data reporting error.

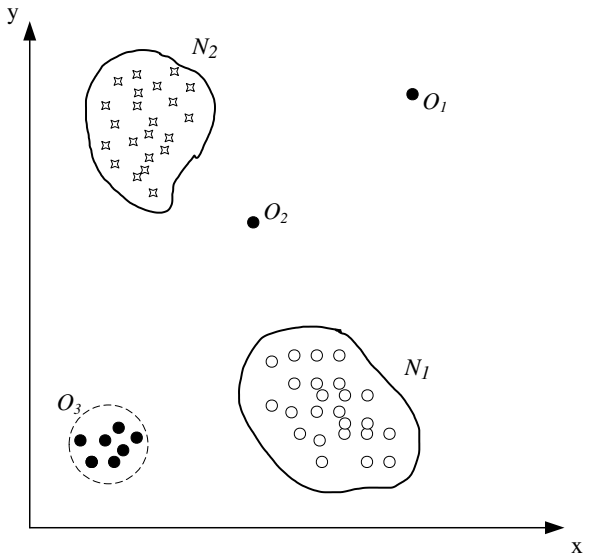


Fig. 2.13 Example of outliers in a 2-D dataset.

There are many ways that outliers can be introduced into the data, but they all share the trait of being intriguing to the researcher. Outlier identification differs from noise removal [113] and noise accommodation [114], which address unwelcome disturbance in the data, in part because of the significance of outliers. Data noise is an obstacle to data analysis because it lacks actual importance on its own. Before performing any data analysis on the data, it is necessary to eliminate any undesirable objects, which is what motivates noise removal. The term *noise accommodation* describes the process of protecting a statistical model's prediction from anomalous data. Novelty detection ([115], [116] and [117]) is a subject linked to outlier detection that seeks to find previously unnoticed (emergent, novel) trends in the data. Novel patterns differ from outliers in that they are usually integrated into the standard model after being discovered.

One of the main benefits of outlier detection is improved accuracy in robotic systems. Inaccurate sensor measurements can result in incorrect or unexpected behavior by robots, which can be dangerous in particular situations. For example, if a robot is equipped with a faulty sensor that reports incorrect distance measurements, it could collide with obstacles or other objects, causing damage or injury. Outlier detection can help identify and correct these types of errors, resulting in more accurate measurements and more reliable robot behavior.

Outlier detection can also improve the resilience of robotic systems. Robotic systems often operate in dynamic and unpredictable environments, where unexpected events can occur, such as sudden changes in lighting conditions or the presence of unexpected obstacles. In these situations, outlier detection can help robots adapt to changes in their environment by detecting and correcting abnormal sensor measurements. By detecting and correcting outliers,

robotic systems can improve their resilience and continue to operate effectively in changing conditions. There are several techniques for outlier detection in sensor measurements, including statistical methods, machine learning algorithms, and rule-based approaches. Statistical methods involve analyzing the distribution of sensor measurements to identify outliers. Machine learning algorithms can learn to detect outliers by training on large datasets of sensor measurements. Rule-based approaches use a set of predefined rules to identify and correct outliers.

This section identifies and discusses the different aspects of outlier detection. As mentioned earlier, a specific formulation of the problem is determined by several different factors such as the input data, the availability (or unavailability) of other resources as well as the constraints and requirements induced by the application domain. This section brings forth the richness in the problem domain and motivates the need for so many diverse techniques.

Input Data

The raw data that an outlier identification method must use to find outliers is a crucial element. According to the author in [118], the input is typically viewed as a group of data objects or data instances (also known as records, points, vectors, patterns, events, cases, samples, observations, or entities). A collection of characteristics can be used to characterize each data instance (also referred to as variable, characteristic, feature, field, or dimension). The data examples may be binary, classified, or continuous in nature. One trait (uni-variate) or several characteristics could be present in each data instance. (multi-variate). All variables in multi-variate data instances may be of the same type or may be a combination of various data types. One crucial finding is that the features used by any outlier identification method do not always relate to the features that can be seen in the provided data collection. In order to work with a set of features that are most likely to distinguish between the typical and outlier behaviors in the data, several techniques employ pre-processing techniques like feature extraction [119], or construct more complex features from the observed features [120]. Finding the best collection of features that will enable the algorithm to produce the best results in terms of precision and computational speed is a crucial challenge for any outlier identification method. On the basis of the structure existing among the data instances, input data can also be classified. The majority of the outlier identification methods in use today work with data where no presumed structure exists among the data instances. These statistics are referred to as point data. Additionally, data may be structured in a geographic, linear, or combined manner. For sequential data, each data instance has a specified ordering that ensures it appears in the complete data collection progressively. The most common illustration for this situation is time-series data, which has undergone intensive statistical

analysis with regard to outlier detection (see [121] and [122]). For spatial data, the data instances have a clearly defined spatial framework, making it important to know where one data instance is in relation to another.

Type of supervision

An outlier identification program may also have access to some extra data in addition to the raw data (or observations). One such piece of information that has been widely utilized is an annotated training data collection (primarily by outlier detection techniques based on concepts from machine learning [123] and statistical learning theory [124]). Techniques that entail creating a clear predictive model need a training data collection. A data instance's identifiers indicate whether it is a normal or outlier instance. Three types of outlier identification methods can be made based on how much these identifiers are used.

Supervised outlier detection techniques Such methods presuppose the existence of a training data collection with annotated examples for both the normal and outlier classes. In such a situation, the typical strategy is to create predictive models for both the standard and outlier groups. The class to which any unknown data instance corresponds is determined by comparison to the two models. Accurate models can be created because supervised outlier identification methods explicitly distinguish between typical and outlier behavior. This has the disadvantage that labeled training data may be extremely costly to acquire. Obtaining the labeled training data collection takes a lot of work because labeling is frequently done mechanically by a human expert. To create a completely annotated training data set, some methods artificially introduce outliers into a normal data set. They then use supervised outlier detection techniques to find outliers in test data [125].

Semi-Supervised outlier detection techniques Such methods presuppose the existence of named examples for just one class. The collection of identifiers for other groups is frequently challenging. Such techniques typically take the strategy of modeling only the classes that are currently accessible and designating any test instance that does not match this model as belonging to the other class. Techniques that presuppose that only the aberrant cases are available for training are not very common. Their low level of adoption is primarily due to the difficulty in obtaining a training data collection that includes all potential aberrant behaviors that might appear in the data. It will be more difficult to spot anomalies in behaviors that don't appear in the training data. For training, the authors in [126] and in [127] have only used anomalous cases. On the other hand, methods that only simulate typical cases during instruction are more widely used. Normal cases are fairly simple to find. It is also simpler to

build sample models for normal behavior from the training data because normal behavior is usually well-defined. This environment is frequently used in harm and defect detection and is very close to novelty detection methods for unsupervised outlier identification (see [115] and [116]).

Unsupervised outlier detection techniques The third group of methods does not rely on the existence of labeled training data in any way. These methods are therefore the most broadly useful. Other suppositions about the data are made by the methods in this group. For instance, parametric statistical methods rely on the assumption that one or both groups of instances have a parametric distribution. Similar to this, many methods start from the premise that typical occurrences occur much more frequently than anomalies. Thus, a pattern that appears frequently is usually regarded as normal, whereas an uncommon event is an outlier. The false alert rate for unsupervised methods is usually greater because the underlying assumptions are frequently incorrect. The aforementioned decision of working modes for any method is governed by label availability. Unsupervised and semi-supervised recognition techniques have typically been used more. Techniques that presume the existence of outlier cases during training are generally not very well-liked. One of the reasons is that it is challenging to obtain a labeled collection of outlying data examples that includes every conceivable kind of outlying behavior. Additionally, the outlying behavior is frequently dynamic in nature (e.g. new types of outliers might arise, for which there is no labeled training data).

Type of outliers

Outliers can be classified into three categories based on their composition and their relation to rest of the data.

`Type I` **Outliers** In a given set of data instances, an individual outlying instance is named a `Type I` outlier. This is the simplest type of outliers and is the focus of majority of existing outlier detection schemes. A data instance is an outlier due to its attribute values which are inconsistent with values taken by normal instances. Techniques that detect `Type I` outliers analyze the relation of an individual instance with respect to rest of the data instances (either in the training data or in the test data). For instance, in credit card fraud detection, each data instance typically represents a credit card transaction. We can assume that the data is defined using only two features time of the day and amount, for simplicity. Figure 2.14 shows a sample plot of the 2D data instances. The curved surface represents the normal region for the

data instances. The three transactions, O_1, O_2 and O_3 lie outside the boundary of the normal regions and hence are `Type I` outliers.

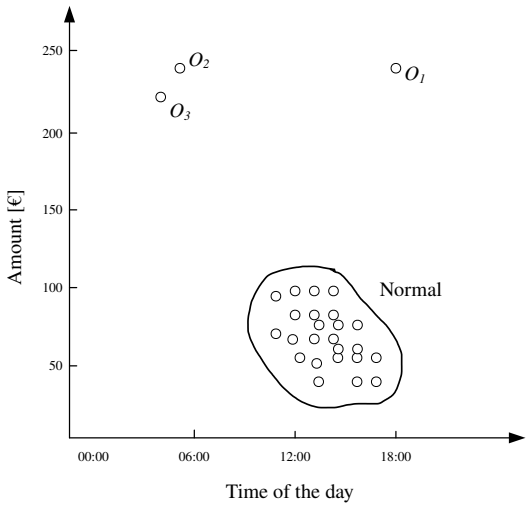


Fig. 2.14 `Type I` outliers O_1, O_2 and O_3 in a 2D credit card transaction dataset. The normal transactions for this data are typically during the day, between 11:00 and 18:00 and range between 10€ to 100€. The outliers O_1, O_2 and O_3 could be fraudulent transactions which are outliers because they occur at an abnormal time and the amount is abnormally large.

`Type II` **Outliers** These outliers are brought about by the appearance of a single data instance in a particular setting in the provided data. These outliers are distinct data examples, just like `Type I` outliers. A `Type II` outlier may not be an outlier in a different setting, which is the distinction. Consequently, `Type II` outliers are described in relation to a situation. The dataset's structure infers the concept of a context, which must be stated as part of the issue formulation. A context identifies a data instance's immediate surroundings.

`Type II` outliers meet two requirements.

- The underlying data is spatial/sequential in character. Contextual and behavioral traits are used to describe each occurrence of the data. The context (or area) of an instance is established using the contextual characteristics, which also describe the location of the instance. The longitude and latitude of a place, for instance, are contextual characteristics in geographic data collections. Alternatively, in a time-series data, time is a contextual characteristic that establishes a particular instance's location within the complete timeline. The non-contextual traits of an entity are defined by the behavioral attributes. The quantity of rainfall at any place, for instance, is a behavioral attribute in a spatial data collection that describes the global average for precipitation.

- The numbers for the behavioral characteristics in a particular situation are used to identify the outlying behavior. In one setting, a data instance might be a `Type II` outlier, whereas in another, the same data instance (in terms of behavioral characteristics) might be regarded as normal.

One such illustration of a temperature time series, depicted in Figure 2.15, shows the region's average monthly temperature over the previous few years. In that location, a temperature of $-5\,^{\circ}$C during the winter (at time t_1) might be considered typical, but the same number during the summer (at time t_2) would be abnormal.

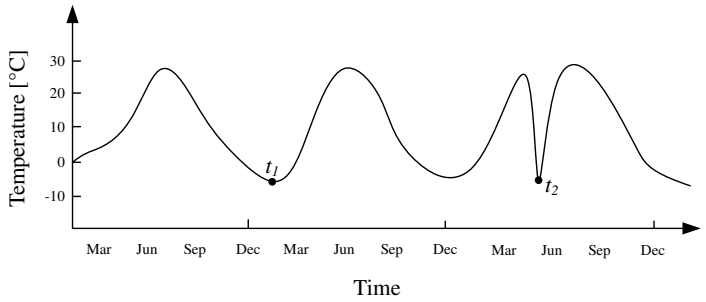


Fig. 2.15 `Type II` outlier t_2 in a temperature time-series. The temperature at time t_1 is same as that at time t_2 but occurs in a different context and thus is not considered as an outlier.

`Type III` **Outliers** These outliers arise from the fact that a portion of the data examples are anomalous relative to the complete data collection. The individual data examples in a `Type III` outlier are not outliers by themselves, but their appearance together as a substructure is anomalous. Only when the data has a spatial or sequential character do `Type III` outliers have any significance. These outliers are either anomalous sub-graphs or sub-sequences that occurs in the data. An illustration of a human ECG output (see [128] for further information) is shown in Figure 2.16. Note that the extended flat line indicates an outlier because the same low value persists for an unusually long period. For sequential data, such as operating system call data and genome sequences, the `Type III` outlier identification issue has been extensively studied. A specific series of operating system calls is regarded as an outlier when analyzing system call statistics. Similarly, outlier identification methods working with images identify areas in the image which are anomalous (`Type III` outliers). It should be mentioned that `Type I` outliers can be detected in any form of data. Sequential or spatial structure in the data is necessary for `Type II` and `Type III` outliers to be detected.

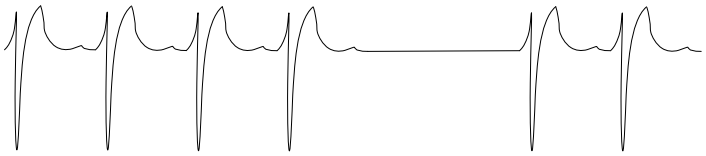


Fig. 2.16 `Type III` outlier in an human electrocardiogram output. The low value in the flat-line also occurs in normal regions of the sequence.

Output of outlier detection

The structure of the outlying patterns identified by the method must meet requirements imposed by the nature of outliers described above. Another prerequisite for any outlier detection method is the way in which the outliers are revealed. Outlier detection methods typically fall into one of the two categories listed below.

Labeling techniques Each test case is given a designation (normal or outlier) by the methods in this group. They act in this way as a categorization system would. The method produces a set of outliers and a set of typical instances when the test input is a set of instances. Such methods have the advantage of giving researchers a precise collection of outliers. These methods have the flaw of not distinguishing between various outliers; no rating of the outliers is given. A zero-one choice is impossible in situations where there is a high degree of certainty that a trend is an outlier. This drives the demand for the scoring-related methods that are covered below.

Scoring techniques Depending on how much a pattern is thought to be an outlier, these methods give it an outlier number. Thus, an ordered collection of outliers is the result of such methods. An researcher can pick the outliers using a cut-off threshold or study the top few outliers. The option of the threshold to pick a group of outliers is the disadvantage of a ranked list of outliers. Choosing this level is frequently difficult and must be done at random. The application area can also set restrictions, such as the intended level of precision and processing speed, in addition to describing the nature of the data and outliers. In domains such as safety-critical systems, the accuracy of the algorithm is a foremost requirement. However, scalable and efficient methods are essential for online systems like network intruder detection systems. The methods frequently need to strike a compromise between the aspects of the answer that depend on accuracy and efficiency. Outlier identification algorithms must handle this issue as privacy protection of data has recently grown to be a significant restriction in a number of areas [129].

2.3 Sensor data generation

Sensor data generation is a critical aspect of resilient robot perception, where accurate and reliable sensor data is essential for robots to perform their intended tasks. In this section we provide an overview, analyzing traditional techniques and deep learning-based methodologies to generate synthetic sensor data.

In traditional techniques, sensor data is generated by using various hardware sensors, such as cameras, LiDAR, and ultrasonic sensors, to capture the environment's information. This data is then processed using classical signal processing techniques, such as filtering and feature extraction, to extract relevant information for the robot's perception. However, gathering sensor data requires hardware (i.e. sensors) and it is time consuming. For that reason, methodologies to generate synthetic sensor data are useful to overcome these problems. Generating synthetic sensor data without using hardware, with traditional techniques, involves creating virtual environments and simulating sensor readings based on the physics of the simulated world. However, these techniques require accurate simulation models and may not capture all the complexities and nuances of the real-world environment. However, with recent advancements in deep learning methodologies, there has been a growing interest in using deep neural networks to generate sensor data for robots. Deep learning techniques, such as generative adversarial networks (GANs), can be used to learn the underlying distribution of the sensor data and generate synthetic data that is similar to the real-world data. This can be particularly useful in scenarios where obtaining large amounts of real-world sensor data may be impractical or too costly. The use of deep learning techniques for sensor data generation has the potential to significantly improve the robustness and reliability of robot perception, especially in challenging and dynamic environments. By generating synthetic data, robots can be trained to handle a wide range of scenarios and adapt to new situations quickly. Furthermore, deep learning-based sensor data generation techniques can also help to improve the generalization and transferability of robot perception models, making them more applicable to real-world scenarios.

Overall, the use of classical techniques and deep learning methodologies for sensor data generation can play a crucial role in developing resilient robot perception systems that can operate effectively in various real-world environments.

2.3.1 Traditional techniques for sensor data generation

By building virtual settings and modeling sensor measurements based on the physics of the simulated world, synthetic sensor data can be generated without the use of hardware. Classical methodologies for sensor data generation have been presented in [130], where

sensor models have been improved to generate synthetic data. We can mention some examples of traditional techniques used to generate synthetic sensor data:

- Physics-based simulation: involves simulating the physics of the environment, including the behavior of objects, lighting, and sound. This technique can be used to generate synthetic sensor data for tasks such as object detection, tracking, and navigation. For example, a physics-based simulation can be used to generate synthetic camera images of an environment and simulate the behavior of objects in that environment.
- Computer graphics: these techniques can be used to create virtual environments and objects, which can be rendered to generate synthetic sensor data. This technique can be used to generate synthetic camera images, LiDAR point clouds, and other types of sensor data. For example, a computer graphics technique called ray tracing can be used to generate synthetic camera images that are physically accurate and realistic.
- Procedural generation: this technique involves generating content algorithmically rather than using pre-made assets. This technique can be used to generate synthetic sensor data for tasks such as object detection, tracking, and navigation. For example, procedural generation can be used to generate synthetic LiDAR point clouds that simulate the environment's geometry and objects or range measurements from UWB or UHF-RFID.

Overall, generating synthetic sensor data without using hardware can be a cost-effective and scalable approach for training machine learning models for robotic tasks. These methods might not fully convey the intricacies and nuances of the real-world environment because they depend on correct simulation models.

2.3.2 Deep learning methodologies for sensor data generation

Deep learning techniques have been increasingly used to generate synthetic sensor data without using hardware. These techniques involve training deep neural networks to generate synthetic sensor data based on a learned model of the real-world environment. Among the deep learning techniques used to generate synthetic sensor data, the following should be mentioned:

- Generative Adversarial Networks (GANs): GANs are a type of deep neural network that can generate realistic synthetic data by training a generator network to produce synthetic samples and a discriminator network to differentiate between the synthetic and real-world data. This technique can be used to generate synthetic camera images, LiDAR point clouds, and other types of sensor data.

- Variational Autoencoders (VAEs): VAEs are another type of deep neural network that can generate synthetic data by learning a low-dimensional representation of the real-world data and then generating new data points based on that representation. This technique can be used to generate synthetic camera images, LiDAR point clouds, and other types of sensor data.
- Deep Reinforcement Learning: Deep reinforcement learning techniques can be used to generate synthetic sensor data by training an agent to interact with a simulated environment and generate sensor readings based on the agent's actions. This technique can be used to generate synthetic sensor data for tasks such as navigation and obstacle avoidance.

Deep learning techniques have been exploited for sensor data processing (as in [131] and [132]) and, in the field of sensor data generation, several studies have explored the use of generative models to generate synthetic data that closely resembles real-world sensor data as in [133], [134] and [135]. Furthermore, deep generative models have been studied in [136], [137] and [138] and gave the basis for a new approach to data generation. Deep generation has been exploited in several applications: noise processing [139], image processing [140], generative modeling of images [141], finance [142], dialogue generation [143] and biomedical applications [144]. In the field of sensor data generation, other studies have explored the use of recurrent neural networks (RNNs) for time-series data generation [145], as well as the combination of multiple deep learning models for generating data from multiple sensors using, for example, Generative Adversarial Networks (GANs) as in [146]. In the context of robotics and autonomous systems, deep learning techniques such as Convolutional Neural Networks (CNNs) have been used for 3D localization of RFID antennas [147] and to detect obstacles with Ultra Wide Band (UWB) with Long Short-Term Memory Neural Networks such as in [148]. In this work, we focused on two specific typologies of sensors: UWB that are widely used in robotics for localization and mapping tasks as in [8] and UHF-RFID that have been used in autonomous systems for Simultaneous Localization And Mapping (SLAM) tasks as in [7].

Finally, in [149] Gartner predicted by 2030 most of the data used in AI (and this will include also robotics) will be artificially generated by rules, statistical models, simulations or other techniques as it is depicted in Figure 2.17. Overall, deep learning techniques for generating synthetic sensor data without using hardware have the potential to improve the scalability and flexibility of robotic perception systems. However, these techniques require large amounts of real-world sensor data to train the deep neural networks effectively, and the synthetic data may not fully capture the complexity of the real-world environment if the deep neural networks are not trained properly on large amounts of data.

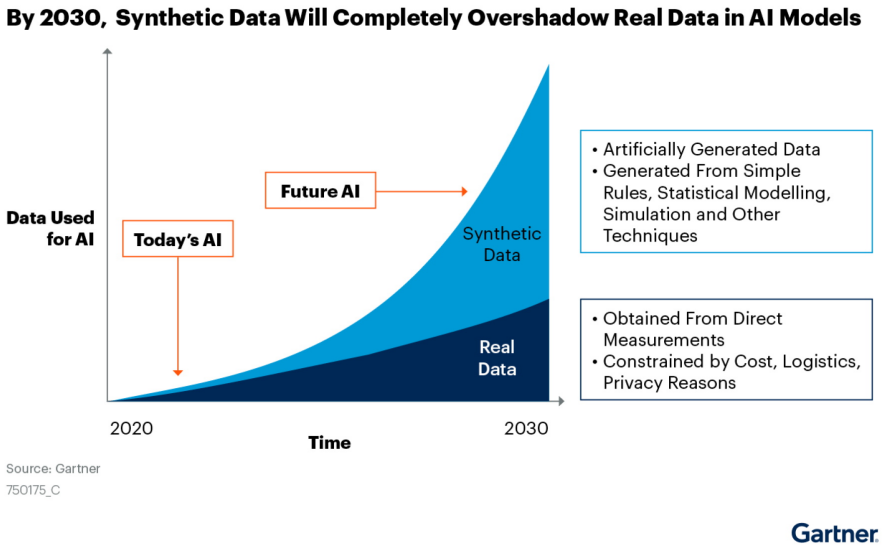


Fig. 2.17 Synthetic data will become the main form of data used in AI, according to Gartner.

2.4 Summary

We have discussed the problems related to resilient robot perception in this chapter. We examined localization, SLAM, VSLAM, and multi-sensor integration as answers to internal issues. Since we are proposing a solution for intelligent robot systems, we focus on the resilient robot perception. Following the findings of the reviewed works, we made the decision to share our research on robot perception resilience using conventional and deep learning methods. We discussed connected research projects based on the common fusion methods used to merge data from various sources. In order to support our choice to select Kalman filters and deep learning techniques as viable alternatives for our application, the benefits and drawbacks of the fusion methods were examined. Its ability to depict incomplete evidence, handle dispute, incorporate reliability, and account for ambiguity from the evidence sources are some of its benefits. In addition to considering fusion methods, the study of cutting-edge works also took into account related works based on the degree at which fusion is carried out. We were able to concentrate on the end applications of the fusion methods and discuss their shortcomings while emphasizing the viewpoints that led to the suggested perception approaches thanks to this alternative revision of the state-of-the-art. One improvement focuses on reducing false positives (outliers), while the other improvement focuses on integrating robustness (resilience) at the sensor fusion level. Information fusion methods within the perception component seek the development of the outcome of the perception task. We describe our multi-sensor fusion contributions in connection to resilience characteristics in the subsequent chapters. We will also go into depth about how the various

modules of the suggested fusion strategy have been implemented. The fusion method will be used in the experimental findings to demonstrate their effectiveness.

Chapter 3

Methodology Overview

The main topics of this chapter are an overview of the designed resilient perception subsystem, the agent, robot and sensor configuration, the multi-sensor fusion proposed architectures and the techniques to generate synthetic sensor data. The sensors presented in this chapter represent the measurement sources of the proposed perception and multi-sensor fusion architectures that will be detailed in Chapter 4 and whose applications will be presented in Chapter 5. The set of sensor configurations are deployed in real autonomous robot contexts. In this chapter we present an overview of the methodology focusing on building a resilient perception architecture. This overview gives the general picture of the goals and requirements of a real resilient perception system helping us to visualise the inputs and outputs of our proposed architecture. In this chapter we also discuss about the importance of synthetic sensor data generation, proposing methodologies to generate real-world sensor data through the use of cutting-edge deep neural network technologies.

Sensor data processing is the first step in every perception system. It involves the sensor configuration, data gathering and data processing. In this chapter, we also present the sensor processing techniques we use to extract useful information from the environment. These techniques focus on the early stages of the perception problem that are usually part of the localization and SLAM components. We will then use the representations and data processing methods presented here to solve localization and SLAM problems within a multi-sensor fusion architecture. In this chapter we also show innovative solutions to localization and SLAM taking into account the possibility of sensor errors/faults or sensor degradation and we will introduce a resilient engine to cope with this kind of problems. We apply state-of-the-art approaches to build localization and SLAM solutions extracting data from several different sensors: cameras, odometry, UWB and UHF-RFID. We will also show that the presented approach can be applied to any range sensor without much effort. The data processing

methods presented in this chapter are common to our multi-sensor fusion approach and are also used in the perception system implementation detailed in Chapter 4.

3.1 Agent, robot and sensor configuration

In this section we introduce the basics to characterize the autonomous systems that we will be using in the methodology description. Specifically, we are referring to agent (Section 3.1.1), robot (Section 3.1.2) and sensor configuration (Section 3.1.3).

3.1.1 Agent

In robotics, an agent refers to an autonomous entity that is capable of perceiving its environment through sensors and acting upon it through effectors. An agent can be any physical or virtual system that is capable of processing sensor inputs and making decisions based on that information. Agents are designed to operate independently and adapt to changing environments by using sensors to gather information about their surroundings and effectors to take actions that modify the environment or their own state. The behavior of an agent can be modeled using a range of techniques, such as decision trees, rule-based systems, neural networks, or reinforcement learning. Agents can be classified based on their level of autonomy, where a fully autonomous agent is capable of operating independently without human intervention, while a semi-autonomous agent requires some level of human input or supervision. Additionally, agents can be classified based on their functionality, such as a cleaning robot or a security robot. Agents are widely used in various robotics applications, such as industrial automation, autonomous vehicles, and service robots. They are also used in many fields outside of robotics, such as artificial intelligence and computer science, to model and simulate complex systems. The agent is, then, an autonomous entity that is capable of perceiving its environment through sensors and acting upon it through effectors, making decisions based on sensor inputs and adapting to changing environments.

In order to characterize the agent to be deployed within the proposed methodology (in 2D), we define it as a dynamic system able to carry multiple sensors. We can then define the agent pose at time t as follows:

$$p_t = \begin{bmatrix} x_{a,t} \\ y_{a,t} \\ \theta_t \end{bmatrix}, \tag{3.1}$$

with $(x_{a,t}, y_{a,t})$ being the agent position and θ_t its orientation as depicted in Figure 3.1. The agent is able to move on the plane and we suppose that there is no *a priori* knowledge of its

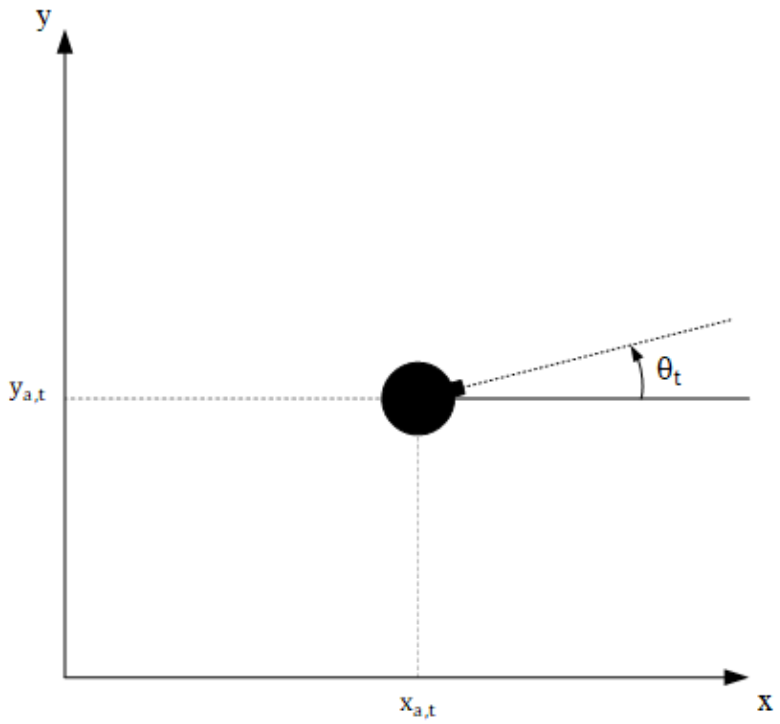


Fig. 3.1 Representation of the agent pose.

dynamics. We made this assumption to let the agent dynamics be as generic as possible. If we suppose to have an odometry system (e.g. visual odometry, see Section 3.1.3 for reference) able to provide the displacements of the agent over time (discretized), expressed as:

$$\Delta_k = \begin{bmatrix} \delta^o_{x,k} \\ \delta^o_{y,k} \\ \delta^o_{\theta,k} \end{bmatrix}, \tag{3.2}$$

we can write the discrete time dynamics of the agent as follows:

$$p_{k+1} = \begin{bmatrix} x_{a,k+1} \\ y_{a,k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_{a,k} + \delta^o_{x,k} \\ y_{a,k} + \delta^o_{y,k} \\ \theta_k + \delta^o_{\theta,k} \end{bmatrix}. \tag{3.3}$$

As stated previously, the agent discrete time dynamics described in Equation (3.3) is referred to the 2D case, however it can be easily extended to 3D. Also, the same equation expresses a generic dynamics and it is not bound to any physical constraint of the agent (i.e. it can be a robot, a human, a vehicle, etc.).

3.1.2 Mobile robot

A mobile robot is a type of robot that is designed to move around and operate in different environments, both indoor and outdoor. Unlike stationary robots that are fixed in one position, mobile robots have the ability to move and navigate through space, typically by using wheels, tracks, legs, or other means of locomotion. Mobile robots can be controlled by a human operator, or they can be programmed to operate autonomously, using sensors to detect and avoid obstacles, and GPS or other localization systems to determine their location. Some examples of mobile robots include autonomous cars, delivery robots, drones, and humanoid robots. Mobile robots have a wide range of applications, including manufacturing, logistics, transportation, agriculture, exploration, and search and rescue. They offer numerous benefits, such as increased efficiency, safety, and precision, and are becoming increasingly common in a variety of industries. In this section we characterize the mobile robot moving on a plane,

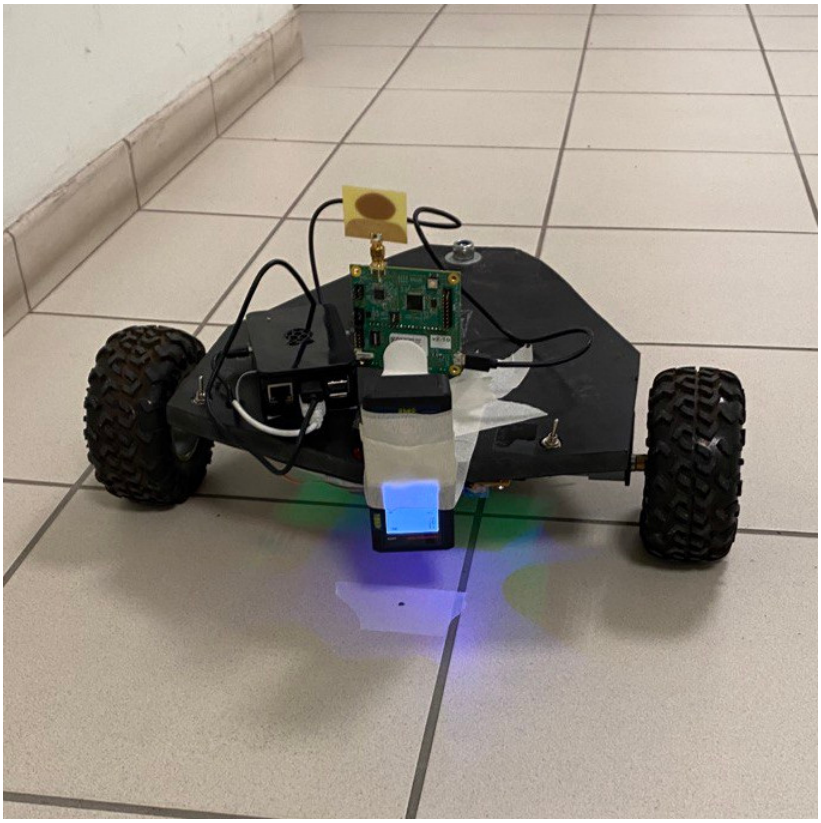


Fig. 3.2 Unicycle-like mobile robot.

from the point of view of its dynamics. Specifically, in order to describe our methodology, we focus on a unicycle-like robot with a differential drive kinematics which operates in an indoor environment (as the one depicted in Figure 3.2). Similarly to the agent pose, described

in the Equation (3.1), we can define the robot pose as follows:

$$p_t = \begin{bmatrix} x_{r,t} \\ y_{r,t} \\ \theta_t \end{bmatrix}, \tag{3.4}$$

with $(x_{r,t}, y_{r,t})$ being the robot position and θ_t the robot orientation. Then, we can write the discrete time dynamics of the robot as:

$$p_{k+1} = \begin{bmatrix} x_{r,k+1} \\ y_{r,k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_{r,k} + u_k \cos(\theta_k) \\ y_{r,k} + u_k \sin(\theta_k) \\ \theta_k + w_k \end{bmatrix}, \tag{3.5}$$

with u_k and w_k being the translation and rotational components of the displacement.

3.1.3 Sensor configuration

One of the most critical components of a robot is its ability to sense its environment and react accordingly. This is where sensors come in. Sensors are devices that measure physical quantities and convert them into electrical signals that can be processed by a robot's control system. In this section, we will explore the various types of sensors that can be used to configure a robot for different applications and that have been selected to develop our methodology. The selection of the right sensor type and configuration is critical in achieving the robot's desired performance. We will cover different sensor types: proprioceptive sensors such as odometry and visual odometry and exteroceptive sensors such as range sensors and vision sensors. Additionally, we will discuss how to mount and interface these sensors with the robot control system. Overall, this section aims to provide a comprehensive understanding of sensor configuration for mobile robots, specifically to build our methodology.

Proprioceptive sensors

A proprioceptive sensor is a type of sensor that measures the internal state of a robot, such as its position, orientation, and velocity, by detecting the changes in the robot's own body. This type of sensor allows the robot to sense its own movements and adjust its actions accordingly. Proprioceptive sensors can include rotary encoders, which measure the angular displacement and speed of a joint, and accelerometers, which measure the acceleration and tilt of the robot's body. In addition, gyroscopes can be used to measure the angular velocity of the robot. These sensors are commonly used in robotic applications such as motion

control, navigation, and robot localization. Proprioceptive sensors are essential in enabling a robot to move accurately and safely, without relying on external references such as GPS or markers. By using proprioceptive sensors, a robot can continuously adjust its position and orientation to achieve its desired motion, making it a valuable tool for a wide range of robotic applications.

In this section, we focus on two typologies of proprioceptive sensors: encoders and visual odometry.

Fig. 3.3 An encoder RS PRO, 500 PPR (pulse per revolution), with 10 mm diameter of the shaft.

Encoders An encoder is a sensor that measures the rotation of a motor or wheel and provides feedback to the robot's control system. Encoders are commonly used in mobile robots to enable precise control of the robot's motion and position. An encoder typically consists of a disk with evenly spaced slots and a light source and detector (see Figure 3.3 for reference). As the disk rotates, the slots interrupt the light beam, generating a series of pulses that correspond to the rotation of the motor or wheel. The frequency and direction of these pulses are used by the robot's control system to determine the speed and position of the robot. Encoders can be used in a variety of ways in mobile robots. For example, they can be used in wheel encoders to provide feedback on the speed and position of the robot's wheels, or in joint encoders to provide feedback on the position of robot arms or other manipulators. By using encoders, mobile robots can navigate accurately and avoid obstacles, perform precise manipulation tasks, and achieve other complex behaviors.

If we take into account the unicycle-like robot described in Section 3.1.2, we can define the relationship between the distance covered by the right and left wheels and the distance covered by the robot and changing in its orientation. So, being d the distance between the

two wheels, from the Equation (3.5) we can write:

$$\begin{aligned} u_k &= \frac{u_{R,k} + u_{L,k}}{2} \\ w_k &= \frac{u_{R,k} - u_{L,k}}{d}, \end{aligned} \tag{3.6}$$

where $(u_{R,k}, u_{L,k})$ is the distance covered in the time interval $(k\,\delta_t, (k+1)\,\delta_t)$ (being δ_t the discretization step) by the right and left wheels. The distance $u_{R,k}$ covered at time step k by the right wheel is related to a noisy encoder reading $u^e_{R,k} = u_{R,k} + n_{R,k}$, where the noise term $n_{R,k}$ is assumed a 0-mean Gaussian random variable with variance given by $K_R|u_{R,k}|$, being K_R a positive constant (similarly, for the left wheel, the variance of the noise term $n_{L,k}$ is $K_L|u_{L,k}|$).

The formulas in (3.6), knowing the encoder readings, can then be used to express the discrete time dynamics of a unicycle-like robot as in (3.5), obtaining:

$$p_{k+1} = \begin{bmatrix} x_{r,k+1} \\ y_{r,k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_{r,k} + \frac{u_{R,k}+u_{L,k}}{2}\cos(\theta_k) \\ y_{r,k} + \frac{u_{R,k}+u_{L,k}}{2}\sin(\theta_k) \\ \theta_k + \frac{u_{R,k}-u_{L,k}}{d} \end{bmatrix}. \tag{3.7}$$

Visual odometry Visual odometry is a technique used in robotics and computer vision that estimates the motion of a robot or a camera based on the changes in the visual input captured by its sensors. The goal of visual odometry is to determine the 3D position and orientation of a camera or robot as it moves through an environment using only visual input. In visual odometry, the camera or robot's motion is estimated by analyzing the changes in the images captured by its sensors over time. This is typically done by tracking features in the images, such as corners or edges, and computing the displacement of these features between frames. By integrating the displacement information over time, the camera or robot's position and orientation can be estimated. Visual odometry is commonly used in robotics applications, such as unmanned aerial vehicles (UAVs), autonomous cars, and mobile robots, to estimate their position and orientation relative to the environment. Visual odometry can also be used in virtual reality and augmented reality applications to track the user's position and orientation in real-time.

The outputs of the visual odometry are generally the camera pose in 3D space and the 3D point cloud. In particular, the camera pose is usually expressed as transformation matrix T_c^f (transformation matrix of the camera relative to a reference frame):

$$T_c^f = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}, \tag{3.8}$$

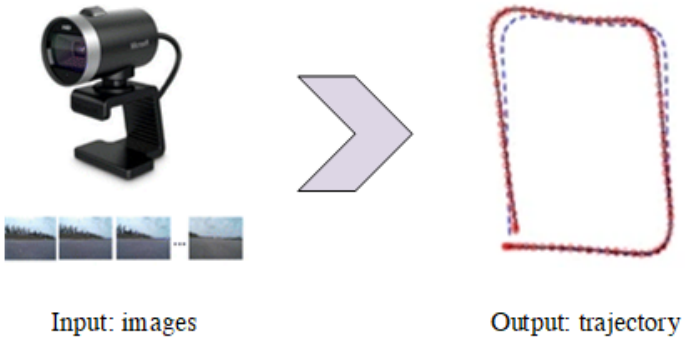


Fig. 3.4 Visual odometry takes images as inputs and generates a camera trajectory as output.

where R is a 3×3 rotation matrix representing the orientation of the camera and t is a 3×1 translation vector representing the position of the camera. The rotation matrix is expressed as:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{3.9}$$

where r_{ij} denotes the i-th row and j-th column element of the rotation matrix. The translation vector t can be expressed as:

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{3.10}$$

where t_x, t_y, t_z represent the elements of the translation vector. The transformation matrix in Equation (3.8) can be then expressed as:

$$T_c^f = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.11}$$

Some visual odometry systems also provide the 3D point cloud that is a collection of 3D points in the scene that have been reconstructed from the visual input. Each point can be represented as a vector:

$$\mathbf{p}_{pc} = \begin{bmatrix} x_{pc} \\ y_{pc} \\ z_{pc} \end{bmatrix}. \tag{3.12}$$

Furthermore, from Equation (3.8) we can obtain the components of the agent displacements Δ_k for the Equation (3.2), specifically, for the translational components $\delta^o_{x,k}$ and $\delta^o_{y,k}$ we have:

$$\begin{bmatrix} \delta^o_{x,k} \\ \delta^o_{y,k} \end{bmatrix} = \begin{bmatrix} t_{x,k} - t_{x,k-1} \\ t_{y,k} - t_{y,k-1} \end{bmatrix}, \tag{3.13}$$

where $t_{x,k}$ is the translation vector representing the camera x position at time k and $t_{x,k-1}$ is the translation vector representing the camera x position at time $k-1$. The same applies to the y components of the translation vector. Finally, in order to determine the rotational component $\delta^o_{\theta,k}$, we have to derive the yaw angle from Equation (3.9):

$$\text{yaw } (\psi) = \text{atan2}(r_{21}, r_{11}), \tag{3.14}$$

and we can write:

$$\delta^o_{\theta,k} = \psi_k - \psi_{k-1}. \tag{3.15}$$

Equation (3.3) can be then expressed as follows:

$$p_{k+1} = \begin{bmatrix} x_{a,k+1} \\ y_{a,k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_{a,k} + t_{x,k} - t_{x,k-1} \\ y_{a,k} + t_{y,k} - t_{y,k-1} \\ \theta_k + \psi_k - \psi_{k-1} \end{bmatrix}. \tag{3.16}$$

Exteroceptive sensors

An exteroceptive sensor refers to a sensor that collects information about the environment surrounding the robot. These sensors enable the robot to perceive and respond to changes in its surroundings, allowing it to navigate through its environment, avoid obstacles, and interact with objects. Examples of exteroceptive sensors used in robotics include cameras that are used to capture images and videos of the robot's surroundings, allowing it to detect and recognize objects, people, and obstacles. LiDARs are sensors that emit laser beams to measure the distance between the robot and objects in its environment; this information is used to generate a 3D map of the robot's surroundings, which can be used for navigation and obstacle avoidance. Ultrasonic sensors are sensors that use sound waves to detect the distance between the robot and objects in its environment; they are commonly used for obstacle detection and avoidance. Infrared sensors are sensors that detect the presence of infrared radiation, which can be used to detect the heat signatures of objects in the robot's environment. Ultra-Wideband (UWB) sensors use short-range radio waves to measure distance and detect the presence of objects in the environment. UWB sensors can be used for a variety of applications in robotics, including localization and mapping, object detection

and tracking and collision avoidance. UHF RFID (Ultra-High Frequency Radio Frequency Identification) sensors are another type of exteroceptive sensor used in robotics that use radio waves to identify and track objects. UHF RFID sensors consist of a reader and one or more tags, which contain an antenna and a small integrated circuit. The reader emits a radio signal, which is picked up by the antenna in the tag. The tag then responds with a signal that contains information about the object it is attached to, such as its identity.

By using exteroceptive sensors, robots can operate autonomously in a wide range of environments, from factories and warehouses to outdoor environments such as fields and forests. In this section we will focus on three typologies of exteroceptive sensors: UHF-RFID, UWB and cameras[1].

UHF-RFID UHF RFID (Ultra-High Frequency Radio Frequency Identification) sensors are a type of exteroceptive sensor used in robotics that use radio waves to identify and track objects. UHF RFID sensors consist of a reader and one or more tags, which contain an antenna and a small integrated circuit. The reader emits a radio signal, which is picked up by the antenna in the tag. The tag then responds with a signal that contains information about the object it is attached to, such as its identity. UHF RFID sensors are commonly used in robotics for a variety of applications, including:

- Object tracking: UHF RFID sensors can be used to track the movement of objects through a warehouse or manufacturing facility. This information can be used to optimize workflows and improve efficiency.
- Inventory management: UHF RFID sensors can be used to track inventory levels and monitor the location of products in a warehouse or retail environment.
- Asset tracking: UHF RFID sensors can be used to track the location of valuable assets such as tools or equipment. This information can be used to improve asset utilization and prevent loss or theft.

One advantage of UHF RFID sensors is their long-range capability, which allows them to read tags from several meters away. This makes them ideal for applications where the object being tracked is moving through a large area, such as a warehouse or distribution center. However, UHF RFID sensors may not be as accurate as other types of sensors, and they require specialized hardware and software to integrate them into a robotic system. Furthermore, the tags are passive and there is no need for powering them with batteries. Another advantage

[1]Among radiowaves-based sensors (that can be classified as exteroceptive sensors), Bluetooth Low Energy, Wi-Fi, ZigBee, LoRa, radars and mm-Wave radars are also used for positioning.

carried by this technology is that the RFID readers can separate the signals coming from the tags, making them capable of solving the data-association problem intrinsically. We consider

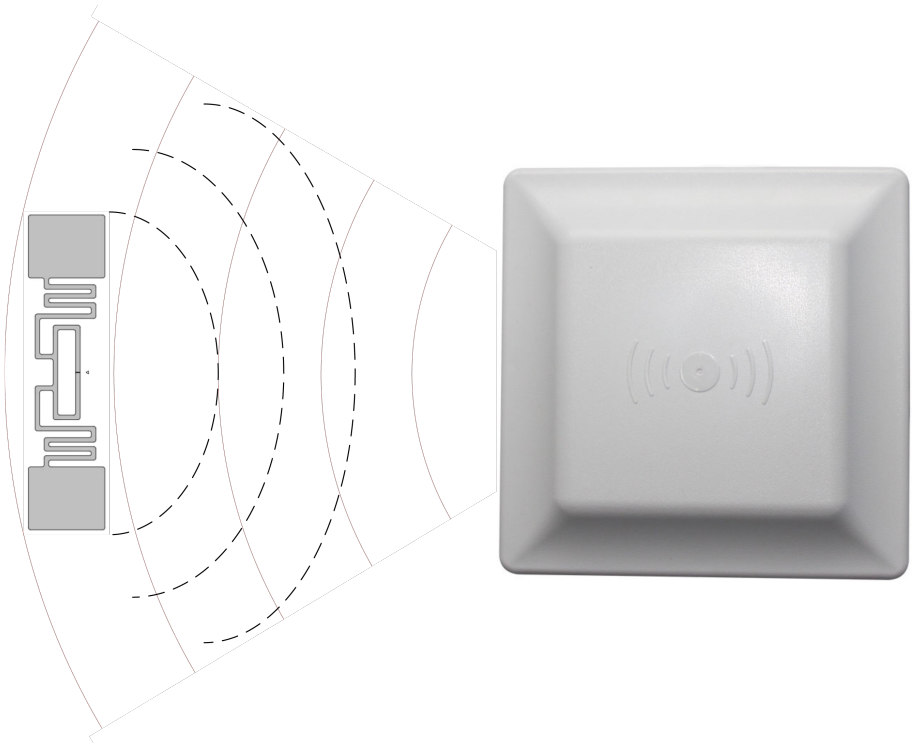


Fig. 3.5 UHF-RFID antenna and tag. At the beginning of the process, the RFID reader (on the right) creates an electromagnetic wave (in red) and transmits it. When the electromagnetic wave meets an RFID tag, the backscatter coupling occurs. With the energy resulting from the coupling, the microchip on the tag (on the left) performs its functions and sends the contents of its memory to the reader, modulating and reflecting the received electromagnetic wave (dashed black lines).

here a reader installed onboard the agent/robot that collects the phases of passive RFID signals backscattered by L tags located in position $(x_{T_1}, y_{T_1}), \cdots, (x_{T_L}, y_{T_L})$. The collected signal at time k from the j^{th} tag is:

$$V_{k,j}(\text{on/off}) = A_{k,j}(\text{on/off}) e^{j\Phi_{k,j}(\text{on/off})}, \tag{3.17}$$

where $A_{k,j}(\text{on/off})$ and $\Phi_{k,j}(\text{on/off})$ are the amplitude and phase of the signal modulated by the tag's binary (on/off) data sequence. In ideal conditions, the phase $\Phi_{k,j}$ accounts for the round trip of wave propagation between the vehicle and the tag. In practical conditions, a commercial RFID reader extracts the differential received signal between the two modulating states of the tag:

$$\xi_{k,j} = \arg(V_{k,j}(\text{on}) - V_{k,j}(\text{off})), \tag{3.18}$$

that includes bias hardware noise and the interference of the signal with the environment. In indoor environments and in presence of scattering objects as furniture, benches, appliances, etc. the signal scattered by the tag feels the effects of the electromagnetic coupling of antennas with the environment, so its intensity and phase differ from that of ideal conditions. In particular there are positions where the signal is so weak that it is not received (i.e. it is below the hardware noise threshold) and positions where the intensity is higher than the noise threshold but the phase is perturbed, i.e. the measured phase $\xi_{k,j}$ differs from the value calculated considering only the round trip path. This effect is particularly noticeable in case of standard tags as inlay tags based on dipole-like antennas and a matching network. They are sensitive to multipath (i.e. the bouncing of the signals from the surfaces of the environment) and to the material of the surface where they are placed on, since they suffer from impedance mismatch and loss of efficiency that affect, moreover, the phase offset. For these reasons, the phase measurement at time k collected by the reader onboard the robot for each tag can be defined as:

$$\phi_k = \mod(-2KD_k + \phi_o + \phi_{m,k} + n_{\phi,k}, 2\pi), \tag{3.19}$$

where $K = 2\pi/\lambda$ (with λ the wavelength of the electromagnetic signal), D_k is the tag-reader distance, ϕ_o is an unknown offset depending on the hardware and $n_{\phi,k}$ is, at each time k, a 0-mean Gaussian noise and $\phi_{m,k}$ is a disturb of the phase that accounts for effects (i.e. multipath) of the environment. The Equation (3.19) could then be used as a model for the UHF-RFID phase measurements, and this will be exploited later in this dissertation.

TriLateration Tags (TLT) In this section, we will discuss about a particular UHF-RFID tag configuration that has been proposed and successfully applied in practice in the context of localization and SLAM problems. Specifically, we observed that the phase difference of the signals of two neighboring RFID tags is a function of the distance between the reader and the tags and this function is single-valued if appropriate conditions are met. Considering the schema shown in Figure 3.6, we have a robot that moves along a straight line x while it receives the signals from two tags deployed on the ceiling at positions $(\pm\frac{d}{2}, 0, Z_T)$. The distance between the tags is d, while the reader-tag distance is D_k, with $k = 1, 2$. The phase of tag k calculated taking into account only the geometric distance is

$$\phi_k = \mod(-2KD_k, 2\pi), \tag{3.20}$$

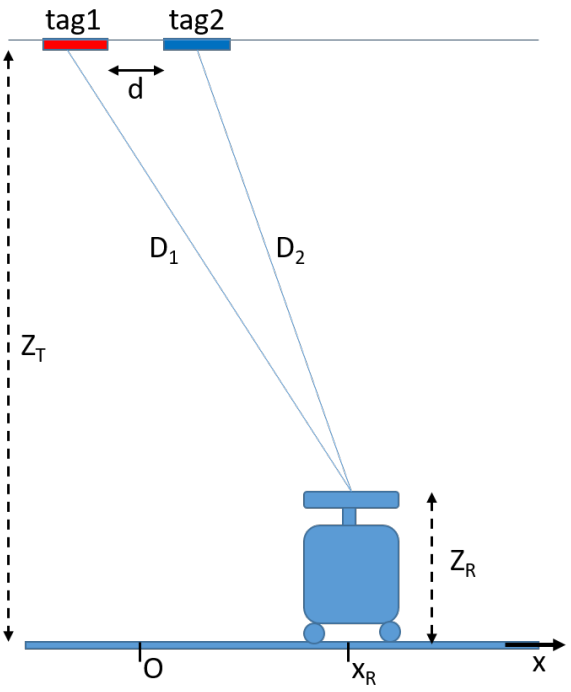


Fig. 3.6 The considered schema for the resilient localization problem.

where $K = 2\pi/\lambda$, $\lambda = 34$ cm is the wavelength of the electromagnetic signal, while the mod function accounts for the periodicity of the phase. The phase difference is calculated as

$$\Delta\phi_{12} = \text{mod}\,(\phi_1 - \phi_2, 2\pi),$$

and is plotted in Figure 3.7 for the cases d =10 cm and d =25 cm, while the reader on board the robot moves along the line $(x_r, 0, Z_R)$ with $-5\,\text{m} < x_r < 5\,\text{m}$ and $Z_T - Z_R = 3\,\text{m}$. It is worth noting that, for $d = 10\,\text{cm}$, $\Delta\phi_{12}$ is a single-valued function of x while, for larger distances, it becomes multi-valued. The single-valued property is limited to the domain where it is defined (i.e. $-5\,\text{m} < x_r < 5\,\text{m}$) and depends on the distance $Z_T - Z_R$, as well as on d. Different combinations of these parameters can be found to satisfy the single-valued property on different domains. Evidently, the single valued function $\Delta\phi_{12}(x)$ allows to localize the robot in the interval $-5\,\text{m} < x_r < 5\,\text{m}$ univocally. That property can be exploited in a 3D domain by means of a suitable deploying of the tags. We chose to deploy tags by grouping them three-by-three as shown in the schema of Figure 3.8, with tags arranged in a radial pattern spaced at an angle of 120°. We call that deployment TriLateration Tag since the short distance d makes the set of three tags a single structure while its use is based on the trilateration principle. Each TLT allows to have three phase differences, namely $\Delta\phi_{12}$, $\Delta\phi_{13}$

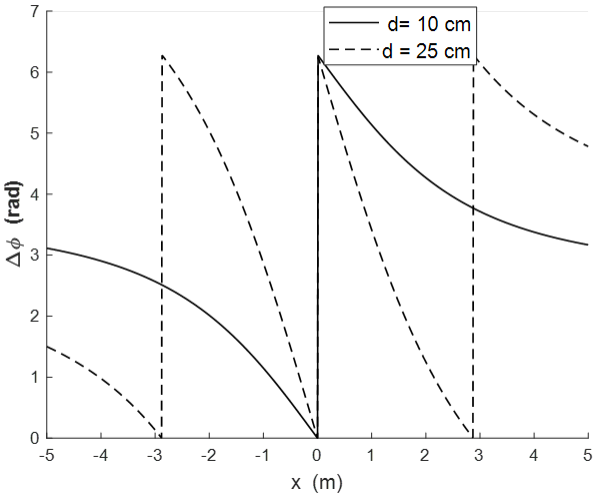


Fig. 3.7 Phase difference $\Delta\phi_{12}$ with $d = 10\,\text{cm}$ and $d = 25\,\text{cm}$, while the reader moves along the line $(x_r, 0, Z_R)$ with $-5\,\text{m} < x_r < 5\,\text{m}$.

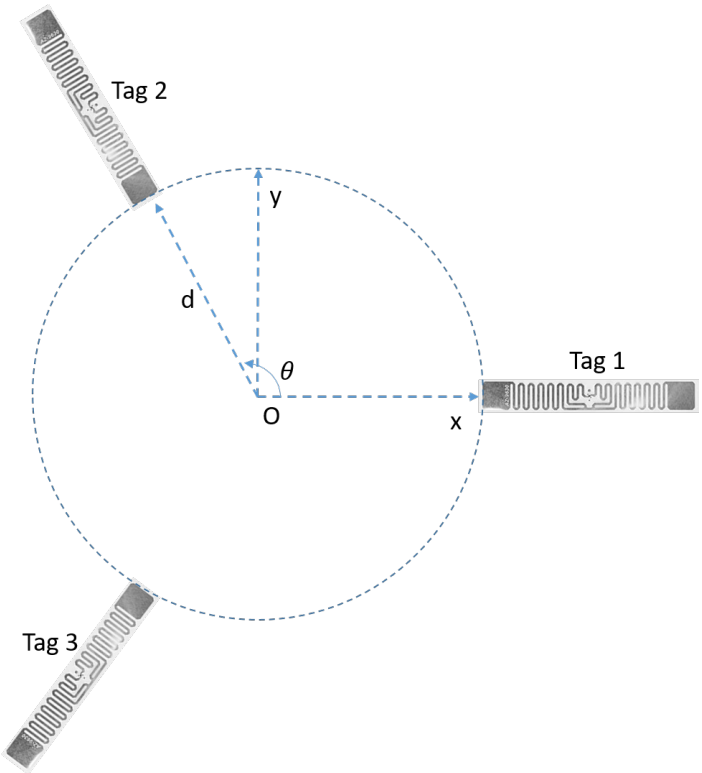


Fig. 3.8 Schema of the TriLateration Tag (TLT).

and $\Delta\phi_{23}$, which constitute a triplet of values that univocally marks each point of the (x,y) plane inside the domain where the TLT has been tuned.

In the case one of the tags composing the TLT is not visible, we consider the TLT as not visible at all. Furthermore, the maximum speed at which this kind of tags can be used depends on the RFID reader sampling time.

This particular configuration for UHF-RFID tags will be exploited in several robot perception applications in the following sections.

UWB Ultra-Wideband (UWB) sensors are a type of exteroceptive sensor used in robotics that use radio waves to measure distance and detect the presence of objects in the environment. UWB sensors operate by emitting short pulses of radio waves at a very high frequency, typically between 3.1 GHz and 10.6 GHz, and measuring the time it takes for the waves to bounce back after they have been reflected by an object. UWB sensors can be used for a variety of applications in robotics, including:

- Localization and mapping: UWB sensors can be used to create detailed maps of indoor environments by measuring the distance between the robot and various objects in the environment. This information can be used to create a 3D map of the environment that the robot can use for navigation.
- Object detection and tracking: UWB sensors can be used to detect the presence of objects in the environment and track their movement. This is particularly useful for robotics applications such as object sorting or autonomous vehicles.
- Collision avoidance: UWB sensors can be used to detect obstacles and other hazards in the environment, allowing the robot to avoid collisions and navigate around them.

One advantage of UWB sensors is their high accuracy and precision, which makes them particularly useful for applications that require precise measurements and real-time data. However, UWB sensors can be expensive and may require specialized hardware and software to integrate them into a robotic system. Figure 3.9 depicts the process of ranging measurement between two UWB antennas that is based on the measurement of the time of flight T_f between the devices and that is defined as follows:

$$T_f = \frac{T_{loop} - T_{reply}}{2}. \tag{3.21}$$

Let us consider a reader installed onboard the agent/robot that collects the range measurements from M UWB antennas located on the ground in position $(x_{u_1}, y_{u_1}), \ldots, (x_{u_M}, y_{u_M})$.

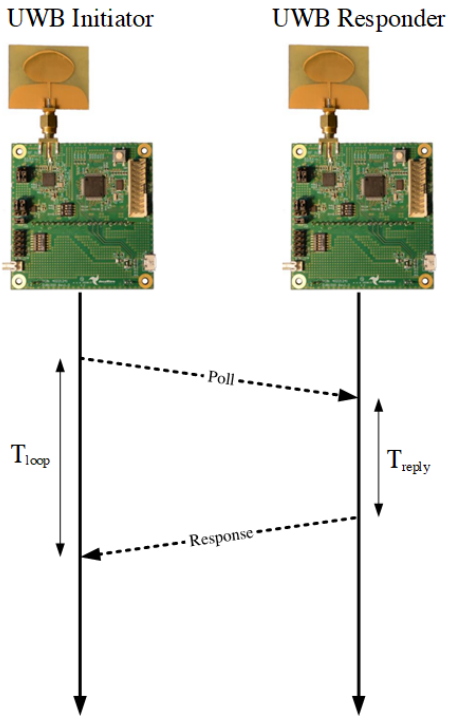


Fig. 3.9 UWB ranging measurement. The ranging is accomplished through Time of Flight (ToF) measurements between the devices; these are used to calculate the roundtrip time of challenge/response packets.

The collected signals at time k from the M antennas are denoted with ρ_k which is defined as follows:

$$\rho_k = [\rho_{1,k} \ \dots \ \rho_{M,k}]^T, \tag{3.22}$$

where $\rho_{1,k}, \dots, \rho_{M,k}$ are the range measurements from the UWB antennas at time k. Each range measurement $\rho_{i,k}$ is affected by a zero-mean Gaussian noise $n_{\rho_i,k}$, $i = 1, \dots, M$, with standard deviation σ_ρ, it is affected by $\eta_{i,k}$ that is the bias due to noise and other errors and it is affected by $\rho_{m,i,k}$ that accounts for multipath effects in the environment. For a UWB antenna i at time k, we can then write:

$$\rho_{i,k} = \sqrt{(x_{u_i} - x_{a,k})^2 + (y_{u_i} - y_{a,k})^2} + \rho_{m,i,k} + n_{\rho_i,k} + \eta_{i,k}, \tag{3.23}$$

where we suppose that the agent is in position $(x_{a,k}, y_{a,k})$ at time k.

Cameras Cameras used for robotics can vary widely depending on the specific application and requirements of the robot. However, there are some general characteristics that are commonly found in cameras used for robotics:

- High resolution: Cameras used for robotics often need to capture high-resolution images or video in order to provide accurate information to the robot's control system.
- Fast frame rate: To capture moving objects or events in real-time, cameras used for robotics typically have a fast frame rate, allowing them to capture multiple frames per second.
- Low latency: Cameras used for robotics need to provide information to the robot's control system as quickly as possible, so low latency is a critical characteristic. This is especially important in applications where the robot needs to react quickly to changes in its environment.
- Wide field of view: Depending on the application, cameras used for robotics may need to capture images or video over a wide field of view in order to provide comprehensive information about the robot's surroundings.
- Robustness: Cameras used for robotics may be subjected to harsh environments or vibrations, so they need to be designed to withstand these conditions.
- Integration with robot control system: To provide the necessary information to the robot's control system, cameras used for robotics must be compatible with the robot's control system and able to communicate with it in real-time.

In particular, there are several types of cameras used for robotics and those can be summarized here:

- 3D cameras: These cameras use depth-sensing technology to create 3D images of the robot's surroundings. This can be useful for tasks such as object recognition, navigation, and obstacle avoidance.
- Thermal cameras: Thermal cameras can detect heat signatures, allowing robots to "see" in low light or no light environments. They can also be used to detect anomalies or changes in temperature, which can be useful for tasks such as monitoring equipment or detecting fires.
- Infrared cameras: Infrared cameras can detect infrared radiation, which can be useful for tasks such as tracking the movement of people or animals.
- Stereo cameras: Stereo cameras use two cameras placed at a distance apart to create a 3D image of the robot's surroundings. This can be useful for tasks such as depth perception, object recognition, and obstacle avoidance.

Overall, cameras are a critical component of robotics and can provide robots with the visual information they need to navigate, interact with their environment, and perform tasks.

Fig. 3.10 The Intel RealSense D435 Camera. It is a 3D depth camera that uses advanced stereo vision technology to capture depth information and enable applications such as augmented reality, virtual reality, and 3D scanning. It features a depth sensor with a range of up to 10 meters, an RGB sensor with a resolution of 1920x1080 at 30fps, and an infrared sensor for improved depth perception in low-light conditions.

In the context of the presented resilient perception subsystem, the cameras, intended as exteroceptive sensors, have been used to perform a vSLAM task based upon the state-of-the-art ORBSLAM2 algorithm, summarized in Section ORB-SLAM2. Figure 3.10 shows a 3D depth camera that uses advanced stereo vision technology to capture depth and RGB images and that has been used in the experiments to validate the methodology presented in this dissertation. Specifically, the ORBSLAM2 algorithm has been used to provide the point cloud (i.e. a map of the environment) as expressed in the formula (3.12).

3.2 Resilient perception subsystem

The ability of robots to perceive their environment accurately is crucial for their successful operation in real-world scenarios. Localization and mapping are two fundamental problems that a robot's perception subsystem must solve to achieve this capability. Localization aims to estimate the robot's position and orientation within its environment, while mapping involves creating a representation of the environment through which the robot is moving. However, these tasks can be challenging due to the uncertainties and dynamic nature of the environment.

Simultaneous Localization and Mapping (SLAM) is a technique that addresses both problems simultaneously by integrating the robot's motion with sensor data to build a map of the environment while localizing the robot within it. The resilience of the robot perception subsystem is critical in real-world scenarios where unexpected events, such as sensor failures or changes in the environment, can occur. This section focuses on the resilient robot perception subsystem, with a particular emphasis on the localization problem and the SLAM problem.

The state-of-the-art techniques and approaches of resilient robot perception have been widely discussed in Chapter 2, to underline the importance of introducing enhancements to the robustness of perception subsystem. In this section, we want to focus on the progress made on resilience in terms of outlier detection and rejection and adaptive algorithms, presenting a methodology based on traditional techniques (e.g. Kalman filter) and deep learning techniques (e.g. Deep Learning) to localization and SLAM perception problems.

3.2.1 Resilient localization

As robots become increasingly integrated into our daily lives, they face the challenge of operating in a variety of environments with varying levels of complexity and uncertainty. Localization, or the ability of a robot to accurately determine its position and orientation in its surroundings, is a critical aspect of robot autonomy and navigation. However, localization can be difficult in environments that are dynamic, noisy, or have limited sensor visibility.

Resilient localization is an emerging area of research that aims to address these challenges by developing robust localization algorithms that can adapt to changing environmental conditions and handle sensor failures or inaccuracies. In this section, we will explore the key concepts and techniques involved in resilient localization for robots, focusing on a setup with a unicycle-like vehicle with a differential drive kinematics with wheel encoders odometry and a system composed of UHF-RFID antenna and RFID tags with structures (denoted in the following as TriLateration Tags (TLT)) comprising three tag antennas close one each

other, displaced in an indoor environment. We will discuss the importance of sensor fusion in developing resilient localization algorithms, as well as the trade-offs between accuracy, complexity, and computational efficiency.

Global resilient localization problem

We are considering here an indoor global robot localization problem where a unicycle-like vehicle moves in an indoor environment and the exteroceptive sensors that are going to be used on the robot are UHF-RFIDs.

Specifically, we exploit the TLT RFID tags configuration, presented in Section TriLateration Tags (TLT), and here we consider them to be placed on the ceiling of the environment. The density of the tags is assumed very low, in such a way that the robot (a unicycle-like vehicle with a differential drive kinematics) rarely detects more than one tag at a time. In order to formulate the global localization problem, we need to describe the robot discrete time dynamics that has been presented in Section 3.1.2 through the Equation (3.5). Since we are considering that the robot is provided with wheel encoders, we can assume that the discrete time dynamics of the robot is the one given in Equation (3.7). Furthermore, a reader is installed on-board the robot and collects the phases of the RFID signals back-scattered by the RFID tags, where the phase measurements are given by the formula in (3.19).

The problem addressed here is the estimation of the robot pose which is assumed completely unknown at the beginning, given the measurements from the encoders and from the UHF-RFIDs. In order to solve the global localization problem, we propose the use of an Extended Kalman Filter to fuse the odometry readings with the phase measurements in order to obtain an accurate pose estimate of the robot, while being robust against disturbances and unmodeled phenomena through a resilient engine. The proposed solution to this problem will be presented in Chapter 4.

3.2.2 Resilient SLAM

Resilient SLAM (Simultaneous Localization and Mapping) is an approach to robotic mapping and navigation that emphasizes the ability of the system to recover from errors or unexpected events. SLAM is a critical function for autonomous robots, enabling them to navigate and map their environments (as described in Section 2.1.4), but traditional SLAM systems can be brittle and prone to failure if the robot encounters unexpected obstacles or the environment changes. Resilient SLAM aims to address these challenges by incorporating techniques such as robust optimization, sensor fusion, and adaptive planning to enable the system to recover from errors and continue operating in challenging environments. By improving the resilience

of SLAM systems, we want to enable more reliable and robust autonomous agents and robots that can operate effectively in real-world settings.

Resilient SLAM using UHF-RFID tags for a unicycle-like robot

In this section we describe a methodology to solve the SLAM problem using UHF-RFID tags, providing the system with a resilient engine in order to cope with outliers in the measurements. The methodology, here exploits a Multi Hypothesis Extended Kalman Filter (MHEKF) which, based on the wheel encoder readings and on the phase measurements coming from a given tag, is able to provide an estimate of the range and of the bearing of that tag with respect to the robot. Once the range and the bearing of the tags is available, standard SLAM approaches can be considered. Since the ID of the tags is available, an EKF-SLAM based algorithm can be adopted as a valid solution to the problem. This approach presents several interesting features that are described below.

First of all the time complexity is low; in fact, even if there is a MHEKF running for each tag, every MHEKF contains a limited number of EKF instances (typically between 10 or 20, depending on the maximum detection range of the reader) and each instance is a 3D EKF (it estimates the range, the bearing and the phase offset of the tag). Another interesting feature is the intrinsic robustness of the approach against disturbances, since, in this case, hypotheses are not pruned and the algorithm is able to restore a good behavior after possible unreliable periods, where strong perturbations could have compromised the effectiveness of the approach, or of part of the approach, for a while. This has been obtained by providing the algorithm with a resilience module, formulated on a robust residual-based adaptive estimation EKF, which may decide to switch off the measurements coming from a given tag if the position estimate of this tag appears not reliable enough, according to defined metrics that will be reported in the following chapters. When the position estimate of the switched off tag returns reliable, this tag may be restored and included in the estimation process. The algorithm may also handle situations where a tag position is changed at some unknown time during the experiment. It is interesting to observe that, even under multipath effects or other perturbations, a wrong but stable estimate of a tag position could provide a virtual landmark which helps in any case the robot pose estimation process. Clearly, in that case, the tag position estimate will be wrong but it can be corrected if the robot reaches an area where the disturbance, causing the wrong tag estimate, decreases. Finally, being the phase offset included in the estimation process, also the problem of an unknown and possibly non constant offset may be handled to some extent by the proposed algorithm.

The system setup considered in this SLAM problem is an indoor environment with a specific number L of RFID tags located on the ceiling, as depicted in Fig. 3.11. The (x, y, z)

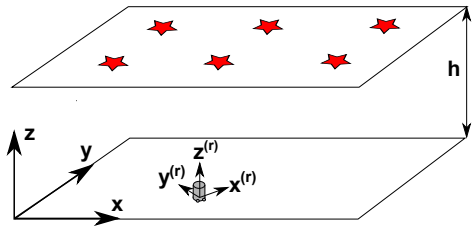


Fig. 3.11 The indoor environment with the robot in its initial position and six tags placed on the ceiling. The triplet (x,y,z) defines the absolute reference frame. The robot reference frame $(x^{(r)},y^{(r)},z^{(r)})$ is also a static frame which is defined by considering the initial pose of the robot.

absolute reference frame is defined by assuming that the floor is the $z=0$ plane, with the z-axis pointing toward the ceiling. For this methodology, everything will be described with respect to this frame. However, in the SLAM algorithm, the robot will reconstruct its path and the tag coordinates with respect to another frame defined with the origin in the initial robot position and with the x axis oriented along the initial motion direction. The robot frame is also a static frame related to the global frame through a roto-translation in the xy plane.

The robot that will be considered is a unicycle-like vehicle with a differential drive kinematics as described by the formulae in Section 3.1.2. The robot carries a UHF-RFID reader that collects the phases of the RFID signals backscattered by L tags located in unknown positions $(x_{T_1},y_{T_1}),\cdots,(x_{T_L},y_{T_L})$, as already mentioned in Section UHF-RFID. If we consider the Equations (3.7) and (3.19) for the discrete time dynamics of the robot and the phase measurements respectively, we can define the range $\rho_{i,k}$ and the bearing $\beta_{i,k}$ of a RFID tag i as follows:

$$\rho_{i,k} = \sqrt{(x_{r,k}-x_{T_i})^2+(y_{r,k}-y_{T_i})^2}, \tag{3.24}$$

$$\beta_{i,k} = \theta_k - \text{atan2}(y_{T_i}-y_{r,k},x_{T_i}-x_{r,k}), \tag{3.25}$$

where (x_{T_i},y_{T_i}) are the unknown coordinates of tag T_i. According to this definition, the distance D_k in (3.19) when the i^{th} tag is observed, is given by $\sqrt{\rho_{i,k}^2+h^2}$, where h is the ceiling height, approximately known. When the robot moves (the tag is assumed fixed in its unknown position (x_{T_i},y_{T_i})), the range $\rho_{i,k}$ and the bearing $\beta_{i,k}$ change over time. Knowing the wheel displacements $u_{R,k}$ and $u_{L,k}$, it is possible to derive the equations which describe the dynamics of the variables $(\rho_{i,k},\beta_{i,k})$. We obtain, after discretization:

$$\rho_{i,k+1} = \rho_{i,k} - u_k\cos(\beta_k), \tag{3.26}$$

$$\beta_{i,k+1} = \beta_{i,k} + \omega_k + \frac{u_k}{\rho_{i,k}}\sin(\beta_{i,k}), \tag{3.27}$$

where u_k and ω_k have been defined in (2.8). Each tag position with respect to the robot is estimated with a Multi-Hypothesis Extended Kalman Filter that fuses the phase measurements with the odometry readings. The objective here, is to solve a Simultaneous Localization and Mapping (SLAM) problem, so that both the robot pose and the tags position (which represent the unknowns of the problem) could be estimated at the same time while resisting the effects of outliers in the measurements. The known quantities in the problem are the encoder readings, the phases of the signal backscattered by the RFID tags on the ceiling and an approximate value of the ceiling height h with respect to the reader on the robot. The solution approach to this problem will be presented in Chapter 4.

Resilient SLAM using TriLateration UHF-RFID tags for a unicycle-like robot

In this section we describe a methodology to solve the SLAM problem using UHF-RFID tags with TriLateration structure, providing the system with a resilient engine in order to cope with outliers in the measurements.

The methodology, here exploits a Multi Hypothesis Extended Kalman Filter (MHEKF) which, based on the wheel encoder readings and on the phase measurements coming from a given tag, is able to provide an estimate of the range and of the bearing of that tag with respect to the robot. Once the range and the bearing of the tags is available, given that the ID of the tags is known, an EKF-SLAM based algorithm can be successfully adopted to solve the problem. As the methodology described in the previous section, the time complexity is low as even if there is a MHEKF running for each tag, every MHEKF contains a limited number of EKF instances (typically between 10 or 20, depending on the maximum detection range) and each instance is a three dimensional EKF (it estimates the range, the bearing and the orientation of the TriLateration Tag with respect to the robot). Another interesting feature of the proposed approach is the resilience against disturbances, since hypotheses are not pruned and the algorithm may be able to restore a good behavior after possible unmodeled perturbations.

This method extends the methodology presented in the previous section by replacing standard RFID tags with TLT structures comprising three tag antennas close one each other that have been already presented in Section TriLateration Tags (TLT). The steady state localization error is significantly lower than the one obtained with uniform mesh of standard tags, deployed with the same density and in principle, if also the orientation of the TLT is considered in the SLAM algorithm, it would be possible to solve a SLAM problem even if only one of these TLT is available. Thus, the proposed approach is particular appealing in large environments, such as warehouses, where tags should be deployed with very low densities. We consider an indoor environment with L RFID TriLateration Tags located on

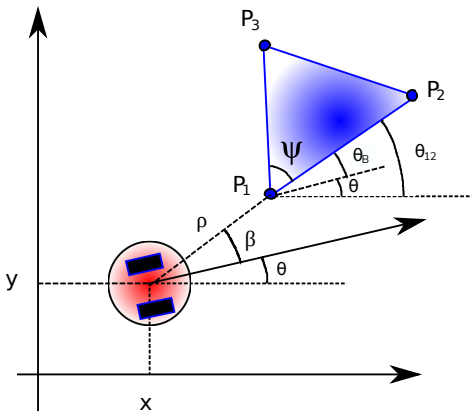


Fig. 3.12 The indoor environment with the robot in position and the TriLateration tag structure with its geometric description.

the ceiling. Each TLT is a structure of the type reported in Figure 3.12. The ceiling height will be denoted by h and will be assumed known. The robot is a unicycle-like vehicle with a differential drive kinematics as the one described with the formulae reported in Section 3.1.2. We consider the Equations (3.7) and (3.19) for the discrete time dynamics of the robot and the phase measurements respectively, considering the tags in a TLT configuration, as we want to derive the discrete time dynamics of the quantities ρ, β and θ_B reported in Figure 3.12. So, if i and j, $i, j \in \{1,2,3\}$, are two antennas in the TLT, let L_{ij} be the known distance between antennas i and j. Let $P_1 = (x_1^{TLT}, y_1^{TLT})$, $P_2 = (x_2^{TLT}, y_2^{TLT})$ and $P_3 = (x_3^{TLT}, y_3^{TLT})$ be the vertexes of the triangle formed by the three antennas of the TLT (see Figure 3.12). We will consider as reference point in the TLT the vertex P_1, and, at time step k, we will define the range and the bearing of the TLT by considering the range ρ_k and the bearing β_k of this vertex with respect to the robot:

$$\rho_k = \sqrt{(x_1^{TLT} - x_{r,k})^2 + (y_1^{TLT} - y_{r,k})^2}, \tag{3.28}$$

$$\beta_k = \text{atan2}(y_1^{TLT} - y_{r,k}, x_1^{TLT} - x_{r,k}) - \theta_k. \tag{3.29}$$

We now introduce also the angle θ_B between the line through P_1 and P_2 and the robot orientation:

$$\theta_{B,k} = \theta_{12} - \theta_k, \tag{3.30}$$

where $\theta_{12} = \text{atan2}(y_2^{TLT} - y_1^{TLT}, x_2^{TLT} - x_1^{TLT})$ is the constant (unknown) angle between the line through P_1 and P_2 and the x-axis of the global reference frame.

It is possible to derive the discrete time dynamics of these quantities (analogously as done in the previous Section) to obtain:

$$\begin{aligned} \rho_{k+1} &= \rho_k - u_k \cos(\beta_k), && (3.31)\\ \beta_{k+1} &= \beta_k + \frac{u_k}{\rho_k}\sin(\beta_k) - \omega_k, && (3.32)\\ \theta_{B,k+1} &= \theta_{B,k} - \omega_k. && (3.33) \end{aligned}$$

The available measurements are the phase readings from the three antennas forming the TLT. To write these measurements, we first compute the distance D_1, D_2 and D_3 of, respectively, antennas 1, 2 and 3 of the TLT from the robot. Exploiting the TLT variables defined so far, we can write:

$$\begin{aligned} D_1 &= \sqrt{\rho^2 + h^2}, && (3.34)\\ D_2 &= \sqrt{\rho^2 + L_{12}^2 + 2\rho L_{12}\cos(\beta - \theta_B) + h^2} && (3.35)\\ D_3 &= \sqrt{\rho^2 + L_{13}^2 + 2\rho L_{13}\cos(\beta - \theta_B - \psi) + h^2}, && (3.36) \end{aligned}$$

where ψ is the (constant and known) angle between the line through P_1 and P_2 and the line through P_1 and P_3 (see Figure 3.12). The phase measurements from the three antennas forming the TLT can then be written as in (3.19), with D_k respectively given by (3.34), (3.35) and (3.36).

It is then possible to estimate the pose of the TLT with respect to the robot, i.e. to estimate the variables (ρ, β, θ_B) through a MHEKF applied to the system with dynamics (3.31)-(3.33) and using the phase measurements coming from the three antennas of the TLT. The output of the MHEKF will be then used as an input for the SLAM algorithm. The details about the solution will be presented in Chapter 4.

Resilient SLAM using UWB for a unicycle-like robot

In this section we describe a methodology to solve the SLAM problem using UWB antennas, providing the system with a resilient engine in order to cope with outliers in the measurements. The proposed SLAM algorithm is based on an always active range and bearing estimation of any responding landmark (i.e. UWB antenna). This estimation can be performed through a 2D Extended Kalman Filter (one for each landmark), fed with the robot odometry and the range measurements. To improve the convergence rate, we propose to use a MHEKF, which comprises a small set of EKF instances, each one initialized with a different possible bearing of the detected landmark. Once the range and the bearing of the responding landmarks is

available, an EKF SLAM algorithm is considered to solve the problem. The MHEKF has the effect of transforming range measurements in range and bearing measurements, and hence to lead the original RO-SLAM problem back to a standard SLAM problem. The proposed EKF-SLAM algorithm is endowed with a resilient module to improve robustness against disturbances. More in detail, if the range and the bearing of a landmark are not considered reliable enough, the associated landmark is not activated or it is switched off if already active. It will be restarted and reinitialized only when its range and bearing estimates will become reliable again. The landmark reinitialization is straightforward since the MHEKF provides both the range and the bearing of the associated landmark. This schema allows to easily face strong perturbations, like the ones indicated above, namely significantly wrong measurements (also at the beginning, due, e.g., to multipath phenomena) or even the shift of a landmark. The decentralized structure of the approach, where the range and the bearing of each landmark are estimated through independent MHEKFs, is particularly effective in reducing the propagation of strong perturbations: as shown through numerical and experimental results, if a landmark is suddenly shifted or strongly perturbed for a while, the proposed approach ignores it and the SLAM algorithm safely proceeds using the other landmarks. Then, when the perturbation is terminated, the landmark is reinitialized and included again in the algorithm.

In the proposed approach, we consider 2D applications (for instance, mobile robots moving on the ground). The height of the landmarks, if different from the height of the robot, is assumed roughly known: this is often the situation in many applications (like, e.g., when the landmarks are placed on the ceiling of an indoor environment). If this is not the case, the MHEKF can be easily extended to estimate also the height of the landmarks.

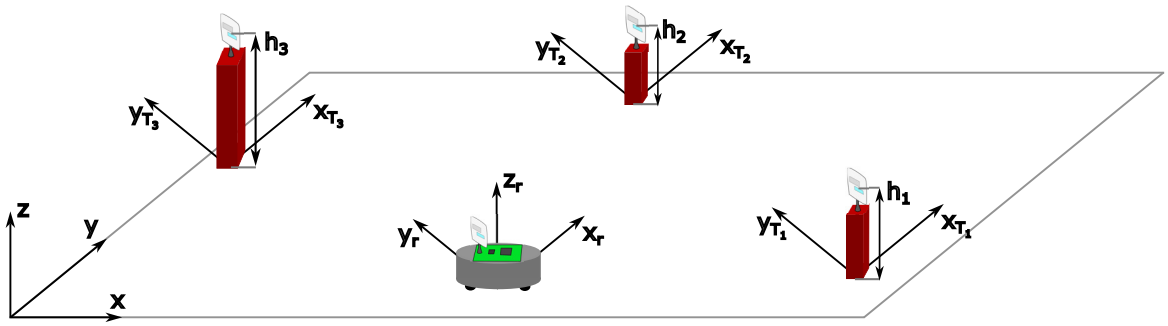


Fig. 3.13 Indoor environment with the robot in its initial position and three UWB antennas placed on pillars at different heights (h_1, h_2 and h_3). The triplet (x,y,z) defines the absolute reference frame. The robot reference frame (x_r,y_r,z_r) is also a static frame which is defined by considering the initial pose of the robot. The position of the three UWB antennas is given by (x_{T_1},y_{T_1}), (x_{T_2},y_{T_2}) and (x_{T_3},y_{T_3}). The height h_i of the antennas, assumed equal to the height of the robot antenna (unless otherwise specified), is also indicated.

For this method, we are going to consider a unicycle-like vehicle with a differential drive kinematics as the one reported in Section 3.1.2 with the equations related to its discrete time dynamics. The robot operates in an indoor environment with L landmarks and we can define $\rho_{i,k}$ as the distance at time k between the robot and landmark i, located in an unknown position (x_{Ti}, y_{Ti}), $i = 1, 2, \ldots, L$ and we can define:

$$z_{i,k} = \rho_{i,k} + n_{i,k}, \tag{3.37}$$

as a measurement of this distance, with $n_{i,k}$ being a 0-mean Gaussian noise with standard deviation σ_{ρ_i}. A schematics of the geometry of the proposed setting with the robot location and the UWB antennas locations is provided in Figure 3.13.

We finally need to define which are the discretized dynamics of the range and bearing of each UWB landmark and if the bearing of the landmark i with respect to the robot at time-step k is:

$$\beta_{i,k} = \text{atan2}(y_{T_i} - y_{r,k}, x_{T_i} - x_{r,k}) - \theta_k, \tag{3.38}$$

we can write the discrete time dynamics of the range and bearing as follows:

$$\begin{aligned} \rho_{i,k+1} &= \rho_{i,k} - u_k \cos(\beta_{i,k}) \\ \beta_{i,k+1} &= \beta_{i,k} + \frac{u_k}{\rho_{i,k}} \sin(\beta_{i,k}) - w_k. \end{aligned} \tag{3.39}$$

In order to solve the presented SLAM problem, the UWB antennas range and bearing will be reconstructed and then a resilient SLAM algorithm will be applied; this will be presented in Chapter 4.

Resilient SLAM using UWB and Visual Odometry for an autonomous agent

In this section we describe a methodology to solve the SLAM problem using UWB antennas and Visual Odometry, providing the system with a resilient engine in order to cope with outliers in the measurements and sensor failures. The proposed SLAM approach is based on an on-line range and bearing estimation of any responding UWB antenna performed through an Extended Kalman Filter (EKF), one for each UWB landmark, fed with the Visual Odometry and the range measurements. The use of Visual Odometry allows to apply the approach in more general contexts where encoder readings are not reliable enough or simply not available (this may occur, e.g., in agricultural applications) to a generic autonomous agent. The range and bearing estimate of a responding UWB antenna, provided by the corresponding EKF, are fused with the VO through an EKF-SLAM algorithm, particularly suited in this context due to the availability of the data association, which allows to realize a

map with the position of the UWB antennas while simultaneously localizing the agent inside this map. The algorithm is resilient in the sense that it is capable of functioning even when the Visual Odometry is not available for a while: in this case the UWB map is frozen and the agent localization is achieved by considering in the main EKF algorithm only the UWB measurements; in this case, the computation times are shorter compared to the ORB-SLAM2 algorithm (that has been used as the Visual Odometry system).

Here then, we propose a method to reconstruct the bearing of a responding landmark in case of range only measurements, thus solving a SLAM problem. Then, the proposed algorithm is endowed with a resilient module to improve robustness against disturbances. More in detail, the resilient module is designed to properly exploit the range and the bearing estimate of the UWB antennas, based on their level of confidence and is also able to understand if it is necessary to reinitialize the estimate of a UWB landmark position (caused, e.g., by a diverging estimate or even by an unknown shift of the UWB antenna). This also allows to overcome the problem of a bad feature initialization, which may seriously afflict other SLAM approaches, where the feature estimate initialization is often performed only once at the beginning. Finally, the proposed algorithm is supervised by a switching observer which modifies the filter structure when a VO failure is detected. Furthermore, the proposed algorithm is capable of working with any number of UWB antennas (at least one should be there if no VO is available) that may also vary during the run.

The system setup considered in this approach is an indoor environment with L UWB antennas located on the ground. The agent carries an UWB antenna at the same height as the other antennas and a camera while moving freely in the environment. The agent has been described in Section 3.1.1 and we are considering the Equation (3.3) for a generic agent moving in the space. The agent considered here also carries an UWB antenna collecting the range measurements from the L UWB antennas located on the ground in unknown positions $(x_{u_1}, y_{u_1}), \ldots, (x_{u_L}, y_{u_L})$. The collected signals at time k from the L antennas are denoted with ρ_k which is defined as follows:

$$\rho_k = [\rho_{1,k} \ \cdots \ \rho_{L,k}]^T, \tag{3.40}$$

where $\rho_{1,k}, \ldots, \rho_{L,k}$ are the range measurements from the UWB antennas at time k. Each range measurement $\rho_{i,k}$ is affected by a zero-mean Gaussian noise $n_{\rho_i,k}$, $i = 1, \ldots, L$, with standard deviation σ_ρ.

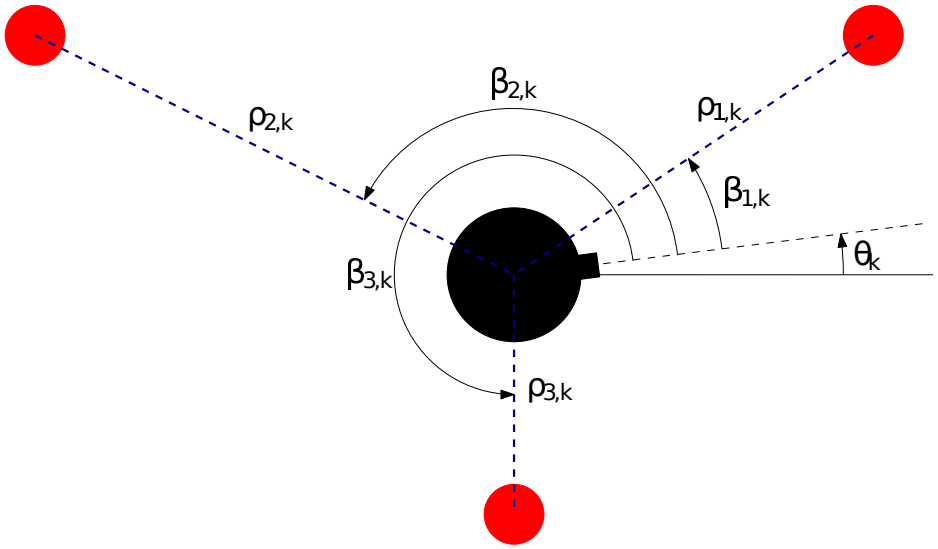


Fig. 3.14 Representation of the system at time-step k. The picture depicts the agent (black) with three UWB antennas (red) placed in its environment.

The range $\rho_{i,k}$ and bearing $\beta_{i,k}$ of the UWB antenna i have been defined as follows:

$$\begin{aligned} \rho_{i,k} &= \sqrt{(x_{a,k}-x_{u_i})^2+(y_{a,k}-y_{u_i})^2} \\ \beta_{i,k} &= \theta_k - \text{atan2}(y_{u_i}-y_{a,k}, x_{u_i}-x_{a,k}), \end{aligned} \tag{3.41}$$

where (x_{u_i}, y_{u_i}) are the coordinates of the UWB antenna u_i. A schema of the system (for $L=3$) is depicted in Figure 3.14, where the black circle represents the agent with its heading, and the three red circles the UWB antennas with their ranges and bearings. We can then write the discretized range and bearing dynamics as follows:

$$\begin{aligned} \rho_{i,k+1} &= \rho_{i,k} - \delta^{vo}_{x,k}\cos(\beta_{i,k}) - \delta^{vo}_{y,k}\sin(\beta_{i,k}) \\ \beta_{i,k+1} &= \beta_{i,k} + \delta^{vo}_{\theta,k} + \frac{\delta^{vo}_{y,k}\cos(\beta_{i,k}) - \delta^{vo}_{x,k}\sin(\beta_{i,k})}{\rho_{i,k}}, \end{aligned} \tag{3.42}$$

where:

$$\begin{bmatrix} \delta^{vo}_{x,k} \\ \delta^{vo}_{y,k} \\ \delta^{vo}_{\theta,k} \end{bmatrix} = \begin{bmatrix} \hat{x}^{vo}_{k+1} - \hat{x}^{vo}_{k} \\ \hat{y}^{vo}_{k+1} - \hat{y}^{vo}_{k} \\ \hat{\theta}^{vo}_{k+1} - \hat{\theta}^{vo}_{k} \end{bmatrix}, \tag{3.43}$$

being $\hat{\mathbf{x}}^{vo}_k = [\hat{x}^{vo}_k, \hat{y}^{vo}_k, \hat{\theta}^{vo}_k]^T$ the estimation of the camera pose at each time-step k, similarly to what we have defined in Section Visual odometry. Summarizing, the problem that we want to solve here is the resilient simultaneous localization and mapping of the agent, so that the agent pose and the antennas position could be estimated at the same time while resisting the

effects of the outliers in the measurements, using UWB readings and Visual Odometry. The proposed solution to this problem will be discussed in detail in Chapter 4.

Resilient SLAM using UWB and Visual Odometry for an autonomous agent exploiting deep learning techniques

In this section we describe a methodology to solve the SLAM problem using UWB antennas and Visual Odometry, providing the system with a deep learning-based approach to model the measurements. The proposed approach is based on an Extended Kalman Filter (EKF) to solve the SLAM problem. The measurement model of the EKF, based on a closed analytic solution, is replaced with a model based on a deep learning architecture. The use of Visual Odometry allows to apply the approach in more general contexts where encoder readings are not reliable enough or not available to a generic autonomous agent. The range and bearing estimate of a responding UWB antenna are fused with the VO through the EKF-SLAM algorithm. The methodology allows to estimate both the agent pose and the UWB antennas position. The algorithm is endowed with a resilient engine able to cope with uncertainties and outliers in the sensor measurements. Also, the system is able to function even when the VO and/or some of the UWB antennas are not available for a specific amount of time.

The system is designed to use a more sophisticated measurement model (based on the training of a Deep Neural Network) that enables the algorithm to calculate better predictions of the measurements, compared to the analytical solution. As stated before, the system is also able to cope with discontinuities in the sensor measurements that makes the system more robust and resilient.

The system setup considered in this approach is an indoor environment with L UWB antennas located on the ground. The agent carries an UWB antenna at the same height as the other antennas and a camera while moving freely in the environment. The agent has been described in Section 3.1.1 and we are considering the Equation (3.3) for a generic agent moving in the space. The agent considered here also carries an UWB antenna collecting the range measurements from the L UWB antennas located on the ground in unknown positions $(x_{u_1}, y_{u_1}), \ldots, (x_{u_L}, y_{u_L})$. The collected signals at time k from the L antennas are denoted with ρ_k which is defined as in Equation 3.40.

The range $\rho_{i,k}$ and bearing $\beta_{i,k}$ of the UWB antenna i have been defined in Equation 3.41. The discretized range and bearing dynamics are defined in Equation 3.42 and in Equation 3.43.

Summarizing, the problem that we want to solve is the resilient simultaneous localization and mapping of the agent, so that the agent pose and the antennas position could be estimated at the same time while resisting the effects of the outliers in the measurements, using UWB

readings and Visual Odometry and while using a more precise measurement model in the EKF-SLAM. The proposed solution to this problem will be discussed in detail in Chapter 4.

3.3 Multi-sensor fusion architectures

Multi-sensor fusion architectures for robotics involve the integration of data from multiple sensors to improve the accuracy, reliability, and robustness of robotic systems. The use of multiple sensors allows the robot to perceive and understand its environment more comprehensively and to make more informed decisions about its actions. There are several different approaches to multi-sensor fusion in robotics, each with its own advantages and disadvantages. In this section, we will present several architectures that have been developed and proposed. In Section 2.1.6 we already mentioned the fusion methodologies: probabilistic, interval calculus, Fuzzy logic, evidence theory and deep learning approaches.

In this section we want to focus on probabilistic approaches to multi-sensor fusion for robots and autonomous agents.

3.3.1 Odometry-RFID sensor fusion

The problem formulated in Section Resilient SLAM using UHF-RFID tags for a unicycle-like robot is solved through an EKF algorithm which uses the range and bearing estimation of each detected UHF-RFID tag provided by a set of MHEKF, one for each tag. The main steps of the approach are summarized as follows.

1. *Initialization*. Initialize for each tag a MHEKF and initialize the EKF SLAM.
2. *Step k: Multi-Hypothesis EKF*. Perform the Multi-Hypothesis EKF for the L UHF-RFID tags, obtaining range and bearing estimates $(\hat{\rho}_{1,k}, \hat{\beta}_{1,k}), \cdots, (\hat{\rho}_{L,k}, \hat{\beta}_{L,k})$.
3. *Step k: Instance instability detection and update*. Perform the *Instability detection step* and update state and covariance according to the *State and covariance update step*.
4. *Step k: Outlier detection*. Perform the *Global outliers detection step*. Update the fault and reliability factors.
5. *Step k: SLAM Prediction*. Perform the *Prediction step* of the EKF algorithm.
6. *Step k: SLAM Correction with robust Kalman gain*. Perform the *Correction step* of the EKF Algorithm with the modified Kalman gain for resilient sensor fusion.

7. Set $k = k+1$ and return to item 2.

□

The sensor fusion is performed through the EKF-SLAM algorithm, fusing odometry readings with the UHF-RFID measurements in order to provide the robot with an estimate of its pose and an estimate of the detected tags position. The details of the multi-sensor fusion architecture will be described in detail in Section 4.1.

3.3.2 Visual-UWB sensor fusion

The problem formulated in Section 3.2.2 for a generic autonomous agent is solved through an EKF algorithm which uses the range and bearing estimation of each UWB antenna provided by a set of EKF, one for each UWB antenna. The main steps of the approach are summarized as follows.

1. *Initialization*. Initialize for each tag a EKF and initialize the EKF SLAM.
2. *Step k: EKF*. Perform the EKF for the L UWB antennas, obtaining range and bearing estimates $(\hat{\rho}_{1,k}, \hat{\beta}_{1,k}), \cdots, (\hat{\rho}_{L,k}, \hat{\beta}_{L,k})$.
3. *Step k: Instance instability detection and update*. Perform the *Instability detection step* and update state and covariance according to the *State and covariance update step*.
4. *Step k: Outlier detection*. Perform the *Global outliers detection step*. Update the fault and reliability factors.
5. *Step k: SLAM Prediction*. Perform the *Prediction step* of the EKF algorithm. If the VO is available, use it, otherwise, use the agent pose estimation from the auxiliary 2D EKF (this check is performed by a switching observer).
6. *Step k: SLAM Correction with robust Kalman gain*. Perform the *Correction step* of the EKF Algorithm with the modified Kalman gain for resilient sensor fusion if the VO is available, otherwise, skip this step (this check is performed by a switching observer).
7. Set $k = k+1$ and return to item 2.

□

The sensor fusion is performed through the EKF-SLAM algorithm, fusing visual odometry with the UWB range measurements in order to provide the robot with an estimate of its pose and an estimate of the detected antennas position. The details of the multi-sensor fusion architecture will be described in detail in Section 4.2.

3.3.3 Multiple vision sensor fusion

A visual-based localization system uses various kinds of frames (*e.g.*, RGB, IR) to estimate the pose of the agent using different techniques, *e.g.*, Visual Odometry, where the agent's motion is inferred by tracking relevant features in the environment. The proposed sensor fusion architecture is based upon the hypothesis that there are several visual odometry systems available for the agent.

The architecture design is based upon the following formal definition.

Definition 1 *A visual position source like visual odometry $\mathscr{S}$ is a system that, equipped with a visual sensor, provides position samples $p_{\mathscr{S},k}$ at a rate $f_{\mathscr{S}}$ and at time-step k.*

The Definition 1 is formulated to cover a wide range of devices and algorithms, as long as they can be interfaced to provide formatted output data for position samples as described in Equation (3.1). The following assumption defines the operational context of this work.

Assumption 1 *There are n visual data sources available, all with the same data rate f_s where each of them provides information on its functioning state at each time-step k: $s_{i,k} \in \{0,1\}$ for $i = 1,\ldots,n$.*

$s_{i,k} = 0$ means that at time k the i-th sensor is not working, whereas $s_{i,k} = 1$ means the opposite. Let $\mathfrak{F} : \mathbb{R} \to \mathbb{R}$ be a filtering operator, and $\mathfrak{B} : \mathbb{R}^{2n} \times \mathbb{R}^n \to \mathbb{R}^2$ be the *blending map*. The main steps of the approach are summarized as follows: each stage integrates sample buffers, whose update operations are not explicitly coded due to space and clarity constraints.

1. *Initialization*. Set $k = 0$ and $i = 0$.
2. *Pre-filtering. Step i, step k*. Perform the pre-processing on the samples by a proper filtering algorithm:

$$\Delta_{i,k} \leftarrow p_{\mathscr{S}_{i,k}} - p_{\mathscr{S}_{i,k-1}}$$
$$\Delta_{i,f,k} \leftarrow \begin{bmatrix} \mathfrak{F}(\Delta_{i,x,k}) \\ \mathfrak{F}(\Delta_{i,y,k}) \end{bmatrix}$$

3. Set $i = i + 1$ and return to item 2 if $i \leq n$, otherwise skip to item 4.
4. *Blending. Step k*. The blending map $\mathfrak{B}$ is used to perform blending as follows:

$$\Delta_{b,k} \leftarrow \mathfrak{B}(\Delta_{1,f,k},\ldots,\Delta_{n,f,k},s_{1,k},\ldots,s_{n,k})$$
$$p_{b,k} \leftarrow \begin{bmatrix} \mathfrak{F}(\Delta_{b,x,k}) \\ \mathfrak{F}(\Delta_{b,y,k}) \end{bmatrix} + p_{b,k-1}$$

5. Set $k = k+1$ and $i = 0$ and return to item 2.

□

The visual sensor fusion is performed using a pre-filtering stage and a blending map-based algorithm, fusing n visual odometry systems in order to provide the agent with an estimate of its pose. The details of the multi-sensor fusion architecture will be described in detail in Section 4.4.

3.3.4 Visual-UWB sensor fusion using deep learning techniques

The problem formulated in Section 3.2.2 for a generic autonomous agent is solved through an EKF algorithm which uses the range and bearing estimation of each UWB antenna provided by a set of EKF, one for each UWB antenna and exploiting a deep learning network to provide a realistic measurement model. The main steps of the approach are summarized as follows.

1. *Training*. Train the deep neural network with a dataset acquired from real UWB sensor measurements.

2. *Initialization*. Initialize for each tag a EKF and initialize the EKF SLAM.

3. *Step k: EKF*. Perform the EKF for the L UWB antennas, obtaining range and bearing estimates $(\hat{\rho}_{1,k}, \hat{\beta}_{1,k}), \cdots, (\hat{\rho}_{L,k}, \hat{\beta}_{L,k})$.

4. *Step k: Instance instability detection and update*. Perform the *Instability detection step* and update state and covariance according to the *State and covariance update step*.

5. *Step k: Outlier detection*. Perform the *Global outliers detection step*. Update the fault and reliability factors.

6. *Step k: SLAM Prediction*. Perform the *Prediction step* of the EKF algorithm. If the VO is available, use it, otherwise, use the agent pose estimation from the auxiliary 2D EKF (this check is performed by a switching observer).

7. *Step k: SLAM Measurement model*. Use the trained deep neural network to predict the sensor measurements, given the inputs (the predicted agent pose and the predicted UWB antennas position).

8. *Step k: SLAM Correction with robust Kalman gain*. Perform the *Correction step* of the EKF Algorithm, using the output of the measurement model and using the modified

Kalman gain for resilient sensor fusion if the VO is available, otherwise, skip this step (this check is performed by a switching observer).

9. Set $k = k + 1$ and return to item 3.

□

The sensor fusion is performed through the EKF-SLAM algorithm, exploiting deep learning techniques to compute the measurements model while fusing visual odometry with the UWB range measurements in order to provide the robot with an estimate of its pose and an estimate of the detected antennas position. The details of the multi-sensor fusion architecture will be described in detail in Section 4.3.

3.4 Synthetic sensor data generation

In order to test the algorithms, researchers and developers need big, meticulously annotated datasets for sensor data. Testing the robot algorithms, such as multi-sensor fusion algorithms, is more effective when they have more varied material. The issue is that compiling and labeling databases that could include a few thousand to tens of millions of components takes a lot of time and is frequently unaffordable. By providing users, developers and researchers with a diverse set of data that accurately represents the real world, synthetic data can resolve private concerns and lessen bias. Synthetic datasets are sometimes superior to real-world data because they are automatically identified and can purposefully include uncommon but important corner instances.

The use of sensors has become increasingly prevalent in various industries, from healthcare to manufacturing. With the proliferation of sensors comes an abundance of data, which can be analyzed to gain insights and make informed decisions. However, the collection and analysis of sensor data can be challenging, especially when dealing with rare events or situations that are difficult to replicate. To overcome these challenges, researchers have turned to synthetic sensor data generation using deep learning techniques. This involves using algorithms to create realistic, simulated sensor data that can be used for a variety of purposes, including training machine learning models, testing sensor systems, and conducting virtual experiments.

In this section we present a classical approach to sensor data acquisition and an effective solution to synthetic data generation exploiting Deep Neural Networks (DNNs) and its application to UHF-RFID and UWB. We want to focus, especially, on the latest advancements in synthetic sensor data generation exploiting deep learning techniques. We will discuss the approach and methodology that have been developed, as well as the challenges and

limitations that still need to be addressed. Furthermore, in Chapter 4 we will examine the potential applications of synthetic sensor data generation on two real experiments involving UltraHigh Frequency Radio Frequency Identification (UHF-RFID) and Ultra Wide Band (UWB) sensors.

3.4.1 Sensor data acquisition

Autonomous robots rely heavily on sensor data to perceive their environment and make decisions on how to act within it. However, the process of acquiring sensor data is not always straightforward and presents several challenges that need to be addressed for successful robot operation. These challenges can arise from a variety of factors, such as sensor noise, limited sensor range, and data fusion from multiple sources. In this section, we will discuss the challenges associated with sensor data acquisition for autonomous robots and explore some of the solutions that have been developed to address them. By understanding these challenges, we can gain insight into the complexities of autonomous robot systems and appreciate the impressive technological advancements that make them possible.

In this section we want to focus on the details about the process of sensor measurements acquisition especially on the UHF-RFID and UWB sensors that have been used to assess the performances of the algorithms for multi-sensor fusion for resilient robot perception.

UHF-RFID sensor measurements acquisition

In this section we describe the measurements acquisition for the UHF-RFID system described in Section UHF-RFID. We mentioned that UHF-RFID sensor consists of a reader and one or more tags which contain an antenna and a small integrated circuit. The reader emits a radio signal which is received by the antenna in the tag that responds with a signal that contains information about the object it is attached to. We performed experiments to acquire this type of sensor data where the reader has been placed on a unicycle-like robot, moving around an indoor environment. During the experiment we acquired the phase of the signal back-scattered from the RFID tag, using the following hardware specification: the reader is the M6e ThingMagic with an antenna with a power of 25 dBm and collects measured data with a rate of 15 Hz while the robot moves with a speed of 0.2 m/s. Measurements have been performed at the frequency of 867 MHz and the tag is type LAB-ID UH107. The row phase measurement from the UHF-RFID is reported in Figure 3.15. The phase of that signal has a periodicity of π, as it can be seen from the figure, and it shows a very noisy behaviour.

The raw signal coming from the sensor should be filtered before using it in any perception algorithm. Specifically, the signal could be processed using a median filter which is a non-

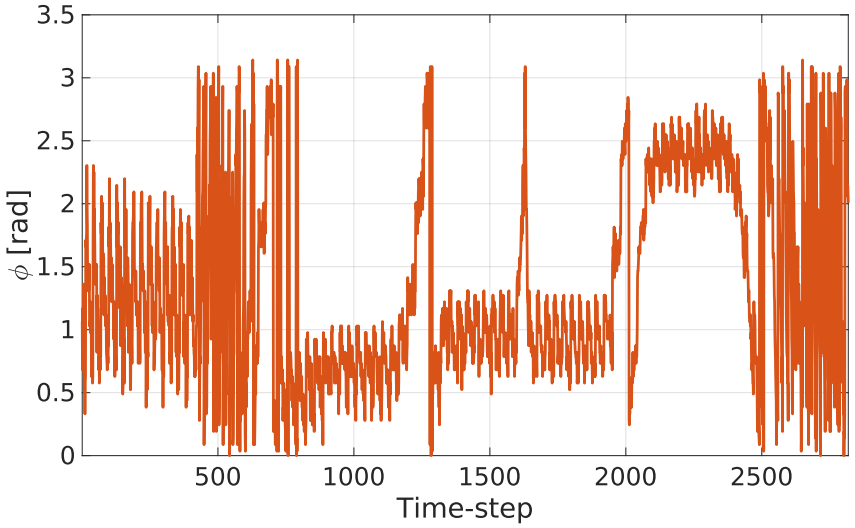


Fig. 3.15 Raw phase measurement from UHF-RFID tag back-scattered signal.

linear digital signal processing technique used to remove noise from the signal. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is the window, which slides, entry by entry, over the entire signal.

The formula for a 1D median filter with a window size of W is:

$$y_i = \text{median}(x_{i-\lfloor \frac{W}{2} \rfloor}, x_{i-\lfloor \frac{W}{2} \rfloor+1}, ..., x_{i+\lfloor \frac{W}{2} \rfloor}), \tag{3.44}$$

where y_i is the filtered value of the i-th sample, x_i is the original value of the i-th sample, and median is the median function.

Furthermore, in order to smooth out the signal and to remove high-frequency noise, a moving average can be also used. The filter works by averaging the values of the input signal over a specific window of time, which can help to reduce the impact of random fluctuations in the signal. The formula for a moving average filter of length N can be expressed mathematically as:

$$y_n = \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i}, \tag{3.45}$$

where x_n represents the input signal to be filtered, y_n represents the output signal after filtering, N is the length of the moving average filter. The formula in (3.45) represents a simple linear time-invariant filter that computes the average of the current input sample and the $N-1$ previous samples. The results of the application of the 1D median filter of window size

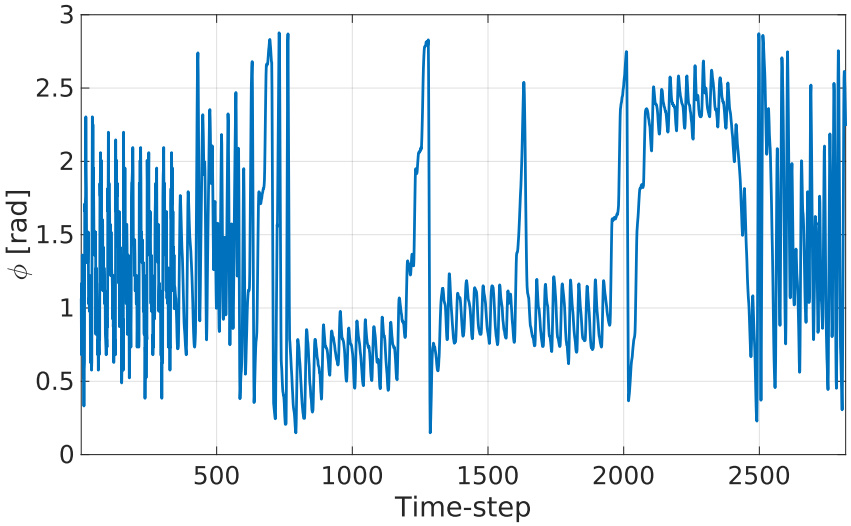


Fig. 3.16 Filtered phase measurement from UHF-RFID tag back-scattered signal.

$W = 6$ and the moving average filter with length $L = 6$ for the signal depicted in Figure 3.15, is reported in Figure 3.16.

UWB sensor measurements acquisition

In this section we describe the measurements acquisition for the UWB system described in Section UWB. We mentioned that UWB sensor uses short-range radio waves to measure distance between objects. We performed experiments to acquire this type of sensor data where the UWB antenna has been placed on a unicycle-like robot, moving around an indoor environment. During the experiment we acquired the raw range measurements with the UWB from a UWB anchor as reported in Figure 3.17 (the raw range measurements are reported in blue).

The raw signal coming from the UWB sensor should be filtered before using it in any perception algorithm. Specifically, the UWB raw signal has been filtered out with a 6^{th}-order 1D median filter (see Equation (3.44), where window size $W = 6$) plus a moving average filter with length $L = 5$ (see Equation (3.45)). Finally, for the UWB sensor system, the measurements have been corrected fitting the bias with a smoothing spline methodology.

The results of the application of the filters are depicted in Figure 3.17 (the filtered range measurements are reported in red). The figure shows how the raw signal has been effectively filtered. The effect of the moving-average filter is particularly visible in plot between the 2000-th and the 3400-th timesteps, where an high frequency noise is present in the data and

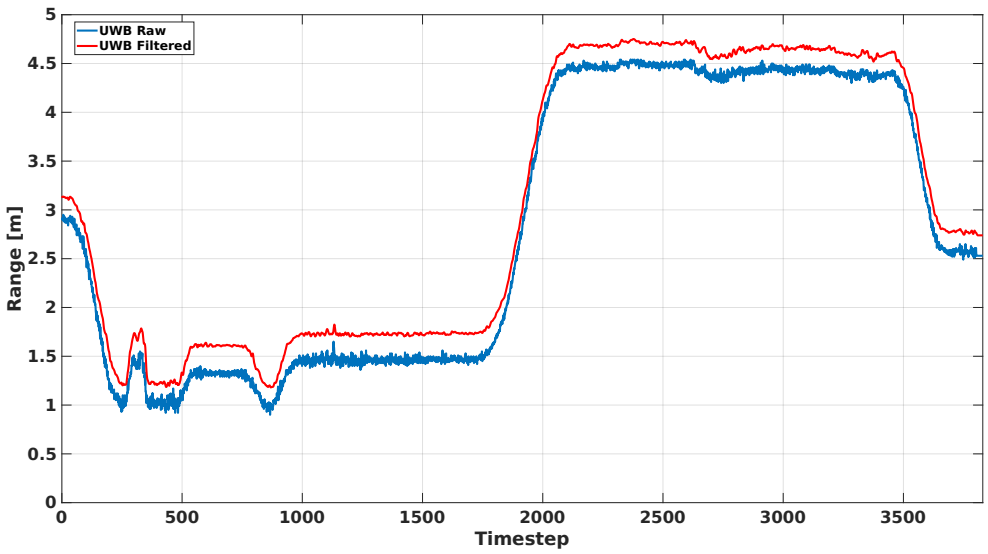


Fig. 3.17 Comparison between raw and filtered UWB measurements for the three UWBs during an experiment. The blue curves represent the raw UWB measurements, while the red curves represent the UWB measurements after applying a 6^{th}-order one-dimensional median filter plus a moving-average filter and the smoothing spline to fit the bias.

is correctly filtered out. Furthermore the smoothing spline bias correction is visible along the plot as a variable displacement between the raw and filtered measurements.

3.4.2 Sensor data deep generation

Sensor data deep generation is the process of synthesizing new data using a combination of sensor inputs. This approach involves training deep learning models to learn the underlying patterns and relationships between different types of data. By combining these different sources of information, it becomes possible to generate more accurate and realistic representations of the real world. The process of sensor data deep generation, presented in this section, has been obtained exploiting Convolutional Recurrent Neural Networks (CRNNs). The model shall be trained on large datasets of multi-modal data to learn the complex patterns and relationships between different types of data. Once trained, the model has been used to generate new, synthetic data that closely resembles the real-world data. One of the primary advantages of sensor data deep generation is its ability to generate more diverse and realistic data compared to traditional methods. This is because the combination of sensor inputs and multi-modal information provides a richer source of information for the deep learning models to learn from, resulting in more accurate and diverse synthetic data. The proposed network architecture has been exploited to generate UWB and UHF-RFID sensor data.

Neural Network Architecture

The proposed architecture for time-series data from sensors is a multi-variate CRNN (Convolutional Recurrent Neural Network) that is an extension of the traditional CRNN architecture designed to handle multiple input signals and variables. Unlike a standard CRNN that takes a single input signal, a multi-variate CRNN takes multiple input signals and combines them to learn a joint representation of the data. The input signals in the proposed multi-variate CRNN can come from a variety of sources, such as multiple sensors, different modalities of data, or multiple channels of information; in our case, in addition to sensor measurements, the other inputs to the CRNN are the robot orientation, position and the UHF-RFID and UWB tag positions. The convolutional layer in the proposed multi-variate CRNN is used to extract features from each of the input signals independently. This layer has been used to capture patterns and structures in each of the input signals that are relevant to the task at hand. The recurrent layer in the proposed network architecture is used to capture the temporal dependencies between the different input signals. This layer allows the network to learn the relationships between the different input signals over time, and to model long-term dependencies between them. By combining the information from multiple input signals, the proposed network can learn to extract a joint representation of the data that is more informative than any individual input signal alone. This leads to improved performance prediction tasks and the novelty presented in this architecture is that the multi-variate CRNN has been extended to a deep generation task. Fig. 3.18 shows the structure of the multi-modal CRNN network; here the inputs to the network have been left generic in order to underline the ability of the network to generalize any kind of sensor data. The CRNN network for n inputs contains the following:

1. n 1D-Convolution layers: Perform 1D convolution (temporal convolution) calculation based on the input data. Each convolution layer accepts an input of dimension N_s (number of time-steps), and the layer consists of 64 output filters with kernel size 2. For each input a convolution layer is instantiated and, during the training phase, the convolution coefficients are trained to compute features.

2. n Activation layers: The activation function for these layers is the ReLU function (Rectified Linear Unit), widely used in neural network and given by:

$$f(x) = \max(0, x)$$

3. n Max pooling layers: Each feature map's dimensionality is decreased while the most crucial data is kept by the spatial pooling. Sub-sampling's goal is to obtain an input

representation by cutting down on its dimensions, which aids in lowering overfitting. The spatial windows size for the Max Pooling layers is 2.

4. *n* Dropout layers: These layers have been added to prevent overfitting.

5. Concatenate layer: This layer concatenates the inputs from the *n* convolution layers, generating a single tensor as a concatenation of all inputs.

6. LSTM layer: The Long Short-Term Memory layer is introduced to simulate long-term dependencies between the various input signals and to learn the relationships between them over time. The dimensionality of the output space is 200.

7. Activation layer: The activation function after the LSTM layer is the tanh function which has been introduced to help mitigating the vanishing gradient problem and stabilizing the training process since the input and output data is normalized between -1 and 1. The tanh function is given by:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

8. Dense layer: The convolutional and LSTM layers are merged to a dense (fully connected) layer. The goal of this layer is to flatten the high-level features that are learned by convolutional and LSTM layers. It has a single output representing the sensor measurement that has to be generated.

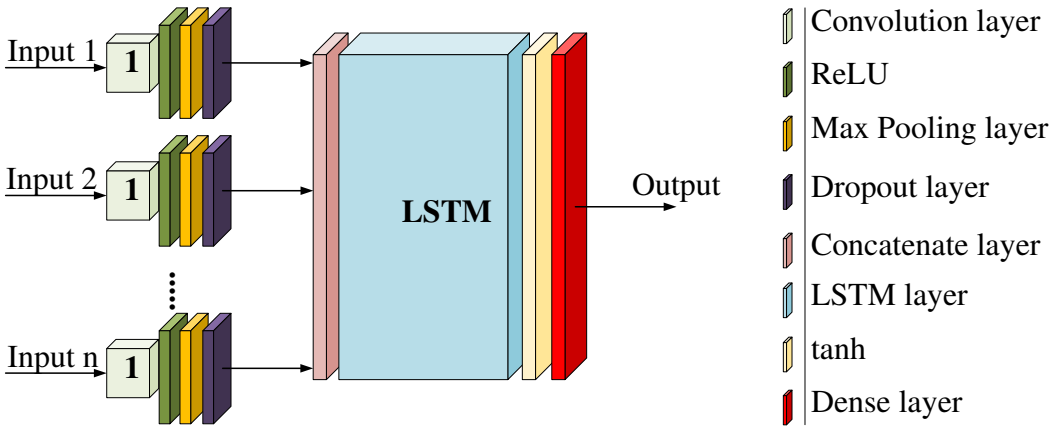


Fig. 3.18 The proposed multi-variate Convolutional Recurrent Neural Network architecture. The network takes *n* inputs, each one with dimension N_s (number of time-steps), and has one single output.

Overfitting

During the training of the CRNN, a method to mitigate the risk of overfitting, which is the phenomenon whereby the model becomes too specialized to the training data and fails to generalize well to new data, has been adopted; the employed technique is dropout, which involves randomly dropping out a proportion of the neurons in each layer during training, thereby forcing the remaining neurons to learn more robust and independent representations of the data, and ultimately improving the model's ability to generalize to new, unseen data. For the presented network architecture, a dropout layer has been introduced for each input with a dropout rate of 0.1; the parameter has been chosen after several trainings, showing the best results.

Loss Function

The goal of training a neural network is to find the parameters (weights and biases) that minimize the loss function, which means the model is able to make accurate predictions on the training data. For regression problems, the mean squared error (MSE) loss function is often used. This function measures the difference between the predicted and true values of the target variable, and penalizes the model for making large prediction errors. Another loss function is the Pearson correlation that is used in machine learning and statistical modeling when the goal is to optimize a model's predictions to maximize the correlation between predicted and actual values. This loss function focuses on the strength of the relationship between the variables, rather than the magnitude of the difference between the predicted and actual values. In this context, we have chosen a combination of MSE and Pearson correlation in order to penalize the model for making large prediction errors and, in the meantime, to maximize the correlation between predicted and actual values. The formula of this loss function is reported in (3.46):

$$f_{\text{loss}} = 1 - \frac{\sum_{i=1}^{m}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{m}(y_i - \bar{y})^2 \sum_{i=1}^{m}(\hat{y}_i - \bar{\hat{y}})^2}} + \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2, \tag{3.46}$$

where m is the number of samples, y_i is the true label for the i-th sample, $\hat{y}_i$ is the predicted label for the i-th sample, $\bar{y}$ is the mean of the true labels, and $\bar{\hat{y}}$ is the mean of the predicted labels. In addition to the loss function, the deep neural network training process also involves an optimizer, which is responsible for updating the model parameters to minimize the loss function. The training on the presented CRNN network has been realized using the Adam optimizer.

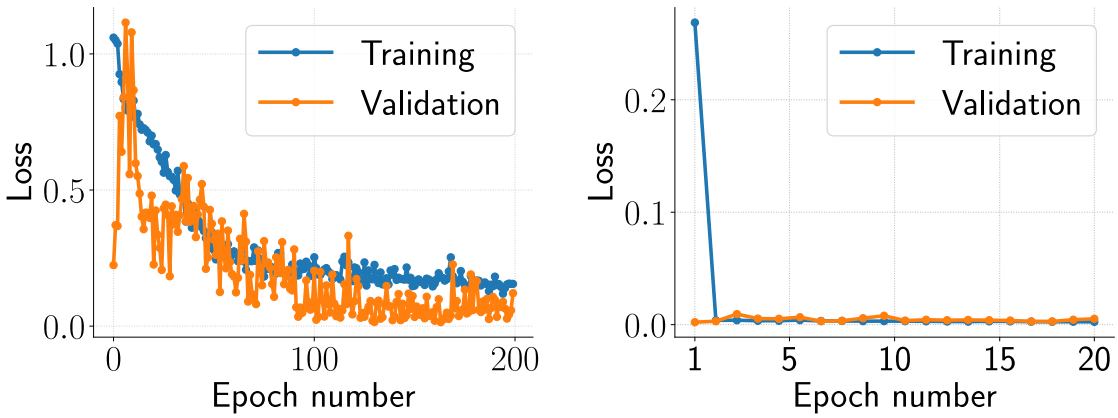


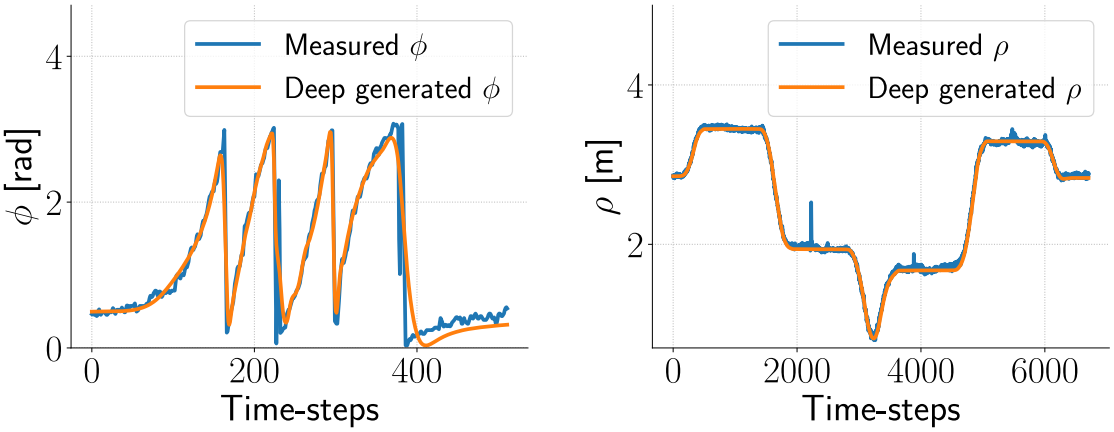
Fig. 3.19 Training and validation loss for (a) UHF-RFID and (b) UWB.

UltraHigh Frequency Radio Frequency Identification

In this experiment, a UHF-RFID reader, with the antenna pointing towards the ceiling, has been mounted on a unicycle-like robot rotating around the axis of the antenna; a RFID passive tag has been mounted on the ceiling in correspondence with the antenna on the robot. The UHF-RFID reader is able to measure the phase shift ϕ in the RFID signal back-scattered by the RFID tag, which is the measurement that we want to generate through the CRNN.

Training results The training data obtained from the acquired sensors consist of multiple acquisitions (> 30) of about 1 min long time-series for the antenna rotation angle θ (as it is estimated from the wheels encoder readings), the phase shift ϕ in the RFID signal back-scattered by an RFID passive tag mounted on the ceiling of the environment in a known position (x_{Tr}, y_{Tr}, z_{Tr}) and the phase offset ϕ_o depending on the hardware. These data have been used to train the CRNN on a Intel Core i7-12700H, 2.3GHz, 14 core and 16GB RAM with an NVIDIA GeForce RTX 3050 Ti GPU. The training time costs around 28 min. The training and validation loss for 200 epochs of the CRNN are depicted in Figure 3.19a. The model is not overfitting for the validation loss is decreasing and the gap between training and validation accuracies are small.

Testing results The trained CRNN network has been tested with real measurements of the antenna rotation angle θ, the RFID tag position (x_{Tr}, y_{Tr}, z_{Tr}) and the phase offset ϕ_o, as inputs; the output of the network is a deep-generated phase ϕ that closely resembles the real-world data, as shown for a single case in Figure 3.20a, where the measured phase is



(a) **(b)**

Fig. 3.20 Deep generated sensor measurements for (a) UHF-RFID and (b) UWB.

reported in blue and the generated phase in orange. The tests have been run on a validation set of real measurements, showing a low average *RMSE* equals to 0.09 rad.

Ultra Wide Band

In this experiment, a UWB antenna has been mounted on a unicycle-like robot moving in a cluttered indoor environment (10 m $\times$ 6 m), where UWB tags have been placed in known positions. The UWB antenna is able to read ranges from each UWB tag.

Training results The training data obtained from the acquired sensors consist of multiple acquisitions ($>$ 50) of about 5 min long time-series for the robot position (x_r, y_r, z_r), the position of the UWB tag located in known position (x_{Tu}, y_{Tu}, z_{Tu}) and the range measurement ρ from UWB antenna to tag. These data has been used to train the CRNN on the same hardware described in Section Training results. The training time costs around 35 min. The training loss for 20 epochs of the CRNN are depicted in Figure 3.19b. The model is not overfitting for the validation loss is decreasing and the gap between training and validation accuracies are small.

Testing results The trained CRNN network has been tested with real measurements of the robot position (x_r, y_r, z_r) and the UWB tag position (x_{Tu}, y_{Tu}, z_{Tu}), as inputs; the output of the network is a deep-generated range ρ that closely resembles the real-world data, as shown for a single case in Figure 3.20b, where the measured range is reported in blue and the

generated one in orange. The tests have been run on a validation set of real measurements, showing a low average *RMSE* equals to 0.06 m.

3.4.3 Measurement data deep generation

Deep learning techniques have brought about a transformative impact on data generation and manipulation in recent times, including the creation of synthetic sensor data. The ability to produce vast amounts of diverse and high-quality data holds immense significance across domains like autonomous driving, robotics, and computer science. The process of generating synthetic sensor data utilizing deep learning methods entails training a model to generate data that closely resembles real-world sensor data. This is accomplished by exposing the model to extensive real-world data, enabling it to discern the inherent patterns and structures within the data. Once trained, the model can generate fresh data that exhibit comparable quality and complexity to the original data, while introducing additional variations and noise to enhance diversity and realism. Noteworthy deep learning techniques such as generative adversarial networks (GANs), variational autoencoders (VAEs), and recurrent neural networks (RNNs) have demonstrated remarkable achievements in generating synthetic data for various sensor types. In this section, deep learning techniques based on Autoregressive Convolutional Recurrent Neural Networks (CRNN) for Multivariate Time Series Prediction, combined with a method based on Denoising Autoencoder (DAE) to learn noise characteristics, have been exploited to generate synthetic sensor data with real world-like noise characteristics. The model has been trained and validated with real data for Ultra Wide Band (UWB) and Ultra High Frequency Radio Frequency Identification (UHF-RFID) sensors. The presented neural network architecture incorporates measurements from sensors and heterogeneous information with the aim of generating synthetic data that can be used to augment real-world data. The deep neural network model presented in this section offers researchers a means to create datasets for algorithm and methodology validation, eliminating the necessity for costly and time-intensive data collection.

Neural network architecture

The neural network is composed of a CRNN similar to that presented in Section 3.4.2 and of a Denoising Autoencoder. Denoising autoencoders belong to a specific type of neural network architecture utilized for unsupervised learning endeavors, particularly in the domains of deep learning and image processing. Their primary purpose is to eliminate noise or corruption from input data and reconstruct the original, uncorrupted data. A denoising autoencoder comprises an encoder and a decoder. The encoder receives the input data, which might be

corrupted or noisy, and maps it to a lower-dimensional latent space representation. This latent representation captures the crucial features of the input data. The decoder takes this latent representation and reconstructs the original data. During the training phase, a denoising autoencoder is exposed to corrupted versions of the input data and is trained to minimize the reconstruction error between the corrupted input and the corresponding clean output. Through this process, the autoencoder learns to extract meaningful features from the corrupted data and generate a denoised output that closely resembles the original, clean data. Denoising autoencoders can be employed for feature learning and dimensionality reduction, whereby the latent space representations learned by the autoencoder can serve as input features for other machine learning models. In this section we used the DAE model with the purpose of learning noise characteristics, the architecture is similar to a standard denoising autoencoder; however, during training, the model is exposed not only to corrupted versions of the input data but also to various levels and types of noise. This exposure allows the network to learn the statistical properties of different noise sources. Specifically, the presented model is able to handle different noise types, including complex and non-Gaussian noise. In order to capture the noise characteristics, we resorted to a DAE model. In most of

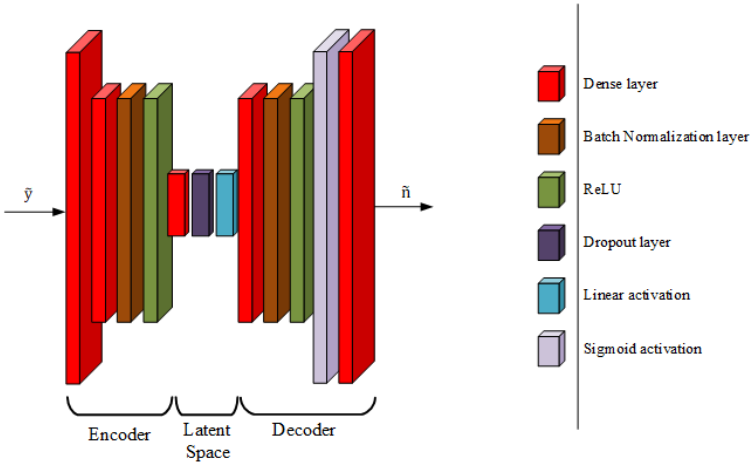


Fig. 3.21 Denoising Autoencoder architecture. The network has 1 multi-dimensional input and 1 multi-dimensional output of the same dimensions.

the cases, the DAE is used to denoise the input data, however, in our approach, we used the DAE to capture the noise statistical properties. The predicted noise will then be added in the overall system to the data predicted by the CRNN network, in order to generate a realistic synthetic sensor measurement.

Figure 3.21 illustrates the structure of the DAE network, where the inputs and outputs have been left generic to emphasize the network's ability to generalize to various types of sensor data. The network has a single multi-dimensional input $\tilde{y}$ of size M_s, the number of

samples from the measurement temporal sequence. The output $\tilde{n}$ has the same dimensionality as the input. The DAE is composed as follows:

1. Encoder dense layers: These layers takes the inputs and flattens them, reducing their dimensions. The resulting outputs of this layer are passed to the batch normalization layer.

2. Batch normalization layers: Batch normalization used here to normalize the activations of each layer by adjusting and scaling them. This helps in addressing the problem of changes in the distribution of layer inputs during the training process (internal covariate shift). This layer has a similar effect to dropout regularization by adding noise to the network, but it is less computationally expensive.

3. ReLU activation layers: The ReLU function (Rectified Linear Unit) is applied to the encoder and decoder layers.

4. Latent space dense layer: This layer compresses the representation of the input data, capturing the essential features and patterns of the input data in a lower-dimensional representation.

5. Latent space dropout layer: The dropout layer prevent overfitting in latent space.

6. Linear activation layer: Equation (3.47) shows the linear activation function, where $f(x)$ represents its output and x its input.

$$f(x) = x. \tag{3.47}$$

   The linear activation function returns the input value as the output without applying any non-linear transformation.

7. Decoder dense layers: These layers has the purpose of reconstructing the original version of the input data from the compressed representation in the latent space.

8. Sigmoid activation layer: The sigmoid activation layer integrates the sigmoid function that is reported in Equation (3.48):

$$f(x) = \frac{1}{1+e^{-x}}, \tag{3.48}$$

   where $f(x)$ is the output of the function, and x is its input. The sigmoid function is a non-linear activation function that maps the input to a range between 0 and 1. It has been integrated in the model in order to introduce non-linearities in the outputs.

Loss function

The proposed DAE network has been designed such that the noise characteristics can be replicated by the model. In order to train it with this capability we have to define a proper loss function that could be able to capture the noise statistical properties. Since the goal of training is to minimize this function, we need to define a loss function that, for the original input and the generated output, could take into account the spatial information and the frequency information.

In order to capture the spatial information during the training phase, we resorted to the MSE that measures the discrepancy between the reconstructed and original inputs. On the other hand, in order to capture the frequency information, we resorted to the PSE that emphasizes the frequency domain, capturing the spectral characteristics and patterns in the data. Furthermore, the combination of MSE and PSE allows for a flexible trade-off between spatial and frequency domains: by adjusting the weight α in the loss function, the DAE behavior can be fine-tuned to prioritize either spatial ($\alpha = 1$) or spectral ($\alpha = 0$) fidelity, providing more control over the reconstruction process. The specific formulation of the chosen loss function, denoted as g_{loss}, is presented in Equation (3.49):

$$g_{\text{loss}} = \alpha \mathsf{MSE} + (1-\alpha)\mathsf{PSE}, \tag{3.49}$$

where $\alpha \in [0,1]$, and MSE is defined in Equation (3.50).

$$\mathsf{MSE} = \frac{1}{s}\sum_{i=1}^{s}(n_i - \hat{n}_i)^2, \tag{3.50}$$

where s represents the number of samples, n_i denotes the true label for the i-th sample, and $\hat{n}_i$ represents the predicted label for the i-th sample.

Finally, PSE is defined in Equation (3.51).

$$\mathsf{PSE} = \frac{1}{F}\sum_{j=1}^{F} |\log_{10}(S_{\text{true}}(j)) - \log_{10}(S_{\text{pred}}(j))|^2, \tag{3.51}$$

where F represents the number of frequency bins, $S_{\text{true}}(j)$ denotes the true power spectrum at frequency bin j, and $S_{\text{pred}}(j)$ represents the predicted power spectrum at frequency bin j, computed using the Fast Fourier Transform as reported in (3.52):

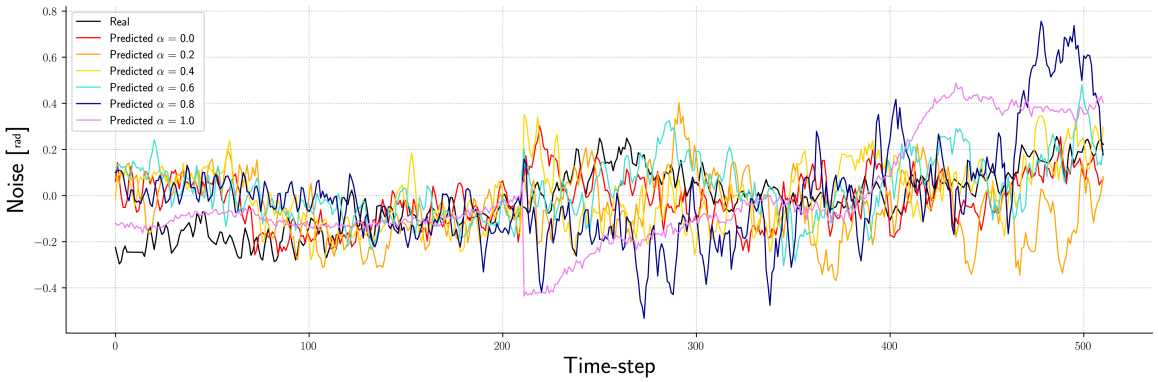
$$\begin{aligned} S_{\text{true}}(j) &= \left|FFT(n_j)\right|^2 \\ S_{\text{pred}}(j) &= \left|FFT(\hat{n}_j)\right|^2. \end{aligned} \tag{3.52}$$

Table 3.1 Noise and power spectrum RMSE average over the validation dataset for different values of α for the UHF-RFID case (lowest RMSE and selected α values in boldface).

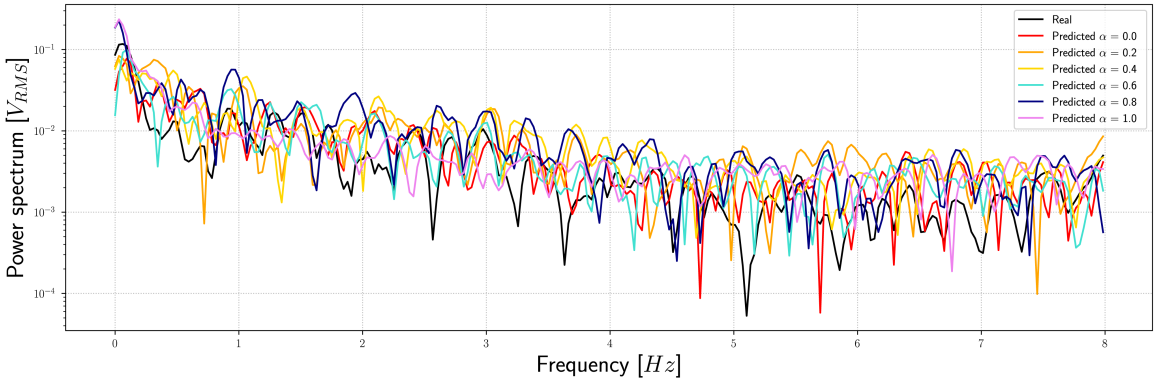
α	Noise RMSE [rad]	Power spectrum RMSE [V_{RMS}]
0.0	0.25956	**0.00900**
0.2	0.25510	0.01037
0.4	0.19041	0.01171
0.6	0.18815	0.01278
0.8	0.18722	0.01294
1.0	**0.16859**	0.01313

Hyper-parameters selection

The hyper-parameter selection aims at optimizing the learning process. The chosen optimizer is the Adam optimizer, whose characteristics and advantages have been already presented and are also satisfying for the DAE network. Also in this case, in order to identify the optimal combination of hyper-parameters, we utilized the grid search method. The intervals and step sizes for the grid search method are as follows: for the number of epochs, the range is set as $[4,50]$ with a step size of 2. The batch size ranges from 2 to 40 with a step size of 2. The learning rate ranges from 0.0001 to 0.05 with a step size of 0.001. For β_1, the range is set as $[0.3,0.99]$ with a step size of 0.01, and for β_2, the range is also $[0.3,0.99]$ with a step size of 0.01. The grid search algorithm has been run on two different datasets for UHF-RFID and for UWB. For the hyper-parameter α, we fixed the best hyper-parameters described so far and analyzed the RMSE between the true and predicted noise signals and the RMSE between the power spectrum of true and predicted noise. In this analysis, we varied the values of α in the range $[0.0,1.0]$ with steps of size 0.2 for both the UHF-RFID and UWB cases. For the UHF-RFID case, Figure 3.22a depicts the real versus predicted noise at different values for α while Figure 3.22b shows the power spectrum for the real and predicted noise, for different values of α for a single generated measurement in the validation dataset. In order to get a better look into the relationship of the α hyper-parameter for the spatial and spectral fidelity trade-off, Table 3.1 provides the values for the average RMSE between true and predicted noise and the average RMSE between the power spectrum of true and predicted noise over all the validation dataset, varying α. From Table 3.1 it can be seen that for the lower bound ($\alpha = 0$), the presented model provides the highest noise RMSE and the lowest power spectrum RMSE: this is consistent with the loss function in Equation 3.49, prioritizing the spectral fidelity on the optimization process. On the other hand, for the upper bound



(a) Real noise (black) and predicted noise (colored) for a UHF-RFID measurement.

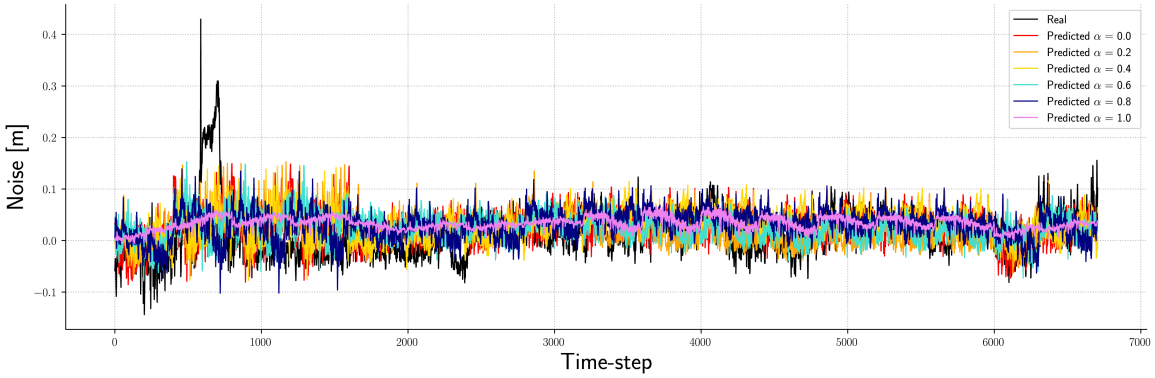


(b) Power spectrum of real noise (black) and predicted noise (colored) for a UHF-RFID measurement.

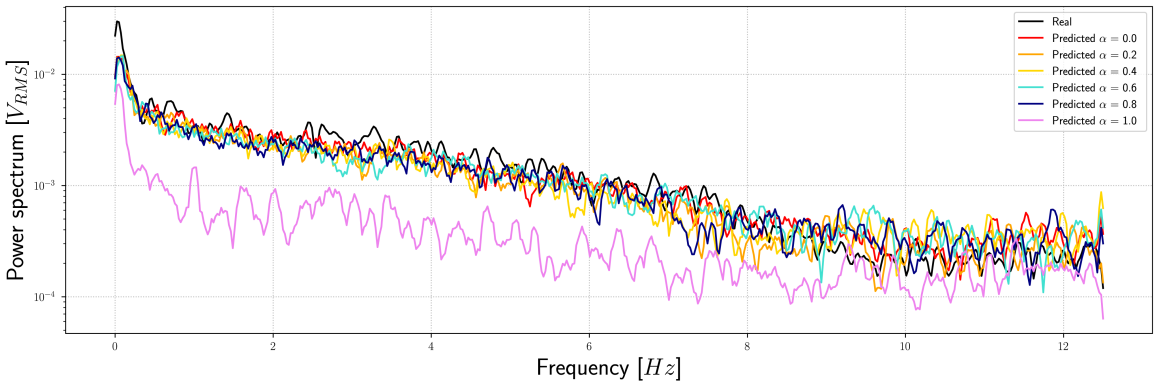
Fig. 3.22 Real and predicted noise (top) and power spectrum of the real and predicted noise (bottom) for UHF-RFID varying the values of the hyper-parameter α for a single measurement in the validation dataset.

($\alpha = 1$), the model shows the lowest noise RMSE and the highest power spectrum RMSE, that is also consistent with Equation 3.49.

For the UWB case, Figure 3.23a depicts the real versus predicted noise at different values for α while Figure 3.23b shows the power spectrum for the real and predicted noise, for different values of α for a single generated measurement in the validation dataset. As per the



(a) Real noise (black) and predicted noise (colored) for a UWB measurement.



(b) Power spectrum of real noise (black) and predicted noise (colored) for a UWB measurement.

Fig. 3.23 Real and predicted noise (top) and power spectrum of the real and predicted noise (bottom) for UWB varying the values of the hyper-parameter α for a single measurement in the validation dataset.

UHF-RFID case, in order to get a better look into the relationship of the α hyper-parameter for the spatial and spectral fidelity trade-off, Table 3.2 provides the values for the average RMSE between true and predicted noise and the average RMSE between the power spectrum of true and predicted noise over all the validation dataset, varying α. As per the UHF-RFID case, from Table 3.2 it can be seen that for the lower bound ($\alpha = 0$), the presented model provides the highest noise RMSE and the lowest power spectrum RMSE: this is consistent with the loss function in Equation 3.49, prioritizing the spectral fidelity on the optimization

Table 3.2 Noise and power spectrum RMSE average over the validation dataset for different values of α for the UWB case (lowest RMSE and selected α values in boldface).

α	Noise RMSE [m]	Power spectrum RMSE [V_{RMS}]
0.0	0.05669	**0.00129**
0.2	0.05558	0.00135
0.4	0.05442	0.00146
0.6	0.05426	0.00151
0.8	0.05342	0.00179
1.0	**0.05215**	0.00250

process. On the other hand, for the upper bound ($\alpha = 1$), the model shows the lowest noise RMSE and the highest power spectrum RMSE, that is also consistent with Equation 3.49.

Finally, the selected hyper-parameters for the two datasets are reported in Table 3.3.

Overfitting

In the DAE we resorted to the dropout methodology in order to prevent the risk of overfitting. For the DAE network architecture, a dropout layer has been integrated in each dense layer of the network, employing a dropout rate of 0.2. The selection of this specific value for the dropout rate was done after running several training sessions, allowing us to choose the best value.

Overall neural network architecture

This section reports the details about the design of the overall neural network architecture. The first component is the convolutional recurrent neural network that aims to capture the spatial and temporal dependencies in the data. The input data for the CRNN, in the case of the UHF-RFID or UWB, can be the antenna position and orientation, the UHF-RFID/UWB tag position and the phase offset (for the only UHF-RFID sensor). This network is the first to be trained and, once trained, it is able to generate the sensor measurements $\tilde{y}$. The generated data resemble the real measurements and they capture the non-linearities in the sensor model (such as, e.g., bias) but the CRNN is not able to model the noise characteristics at this stage. In order to also capture that, we combined the CRNN with a denoising autoencoder. The DAE takes the sensor measurements $\tilde{y}$ (including L samples) generated by the CRNN as an input; these inputs are organized as chunks of samples of size M_s such that each input X_i to the DAE is in the form:

$$X_i = \{\tilde{y}_i, \tilde{y}_{i+1}, \ldots, \tilde{y}_{i+M_s}\}, \tag{3.53}$$

Table 3.3 Selected hyper-parameters for the DAE network.

Hyper-parameter	Value	Sensor type
Epoch	20	UHF-RFID
Batch size	4	UHF-RFID
Learning rate	0.002	UHF-RFID
β_1	0.88	UHF-RFID
β_2	0.998	UHF-RFID
ε	10^{-8}	UHF-RFID
α	0.6	UHF-RFID
Epoch	20	UWB
Batch size	14	UWB
Learning rate	0.002	UWB
β_1	0.9	UWB
β_2	0.997	UWB
ε	10^{-8}	UWB
α	0.4	UWB

with $i \in [1, L - M_s]$ and $\tilde{y}_i$ representing the i-th sample for the sensor measurement signal $\tilde{y}$. The DAE model is trained right after the CRNN: this is an architectural constraint as the DAE input X_i relies upon the generated measurements $\tilde{y}$. Once trained, the DAE model is able to generate the noise signal $\tilde{n}$ that resembles the real noise in terms of spatial and spectral characteristics. At the final stage, the model combines the signal $\tilde{y}$ generated by the CRNN with the noise $\tilde{n}$ generated by the DAE into $\tilde{y}_{nl}$ as follows:

$$\tilde{y}_{nl} = \tilde{y} + \tilde{n}. \tag{3.54}$$

The overall architecture diagram is depicted in Figure 3.24.

Experiments

In this section, the experiments for the UHF-RFID and UWB cases are reported. The overall network architecture is composed by a CRNN and a DAE networks, as presented in Section 3.4.3. The training is divided into two phases: in the first phase, the CRNN is trained with the appropriate inputs and true values y. Once trained, the CRNN network is able to generate the signals $\tilde{y}$ that will be used in the second phase of the training by the DAE network. In that phase, the DAE will take the inputs X_i (as per Equation (3.53)) and the true values n.

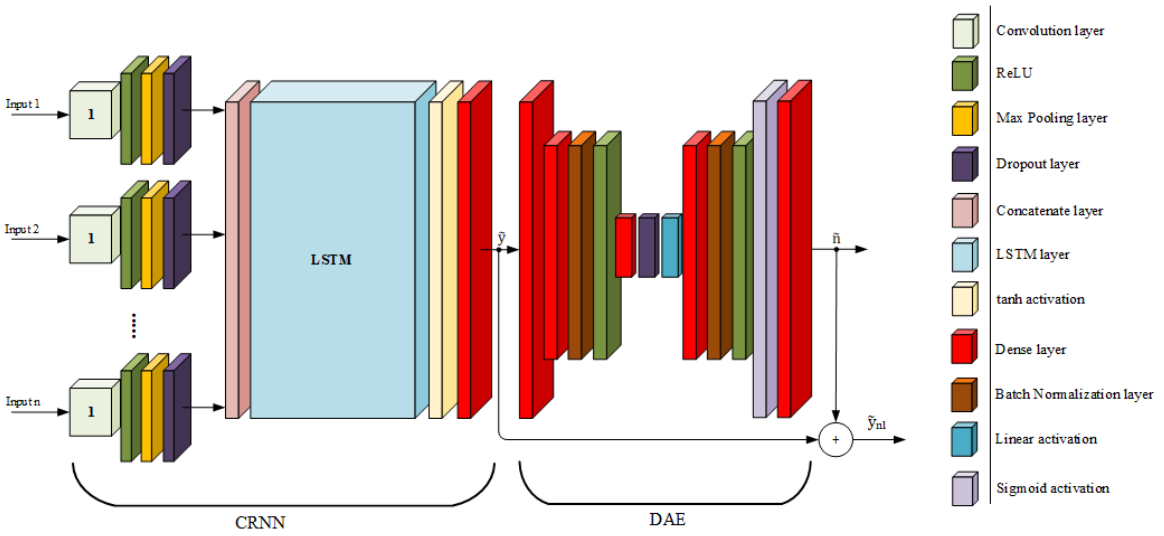


Fig. 3.24 Overall architecture diagram of the presented deep neural network. The model is a combination of CRNN and DAE. The legend on the right reports the layers as they appear from left to right in the left diagram.

In order to ensure a good generalization ability, we designed the training and test samples in order to assure that they have been drawn from the same underlying distribution. Then, we used the 80 % of the data for training and the remaining for evaluating the model's performance for both the CRNN and DAE networks. Furthermore, we captured temporal variability in both training and testing samples and randomly shuffled the training data in order to prevent the CRNN from overfitting to specific temporal patterns present in the training set. We also captured the spatial and spectral variability in the training and validation samples when training the DAE network and randomly shuffled the training data in order to prevent overfitting to specific spatial and spectral patterns present in the training set. Finally, we also used regularization techniques (dropout as mentioned in 3.4.3) in order to reduce the networks' sensitivity to specific training samples and to achieve a more robust generalization capability.

Validation Metrics

In order to evaluate the ability to generate synthetic data that resemble the real sensor measurements, we resorted to two different metrics in order to assess both the spatial and spectral consistency of the generated signals. The MSE between the real and generated signal is used to capture the network ability to generate spatial consistent synthetic measurements as reported in Equation (3.55), where y_i represents the i-th sample of the real sensor measurement

y and $\tilde{y}_{nl,i}$ is the i-th sample of the sensor measurement $\tilde{y}_{nl}$ generated by the network.

$$\mathsf{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \tilde{y}_{nl,i})^2. \tag{3.55}$$

On the other hand, the magnitude-squared coherence (MSC), used as a measure of the similarities in the frequency content of two signals, is selected to assess the network ability to generate spectral consistent synthetic measurements. This metric is reported in Equation (3.56):

$$\mathsf{MSC}(f) = \frac{|G_{y\tilde{y}_{nl}}(f)|^2}{G_{yy}(f)G_{\tilde{y}_{nl}\tilde{y}_{nl}}(f)}, \tag{3.56}$$

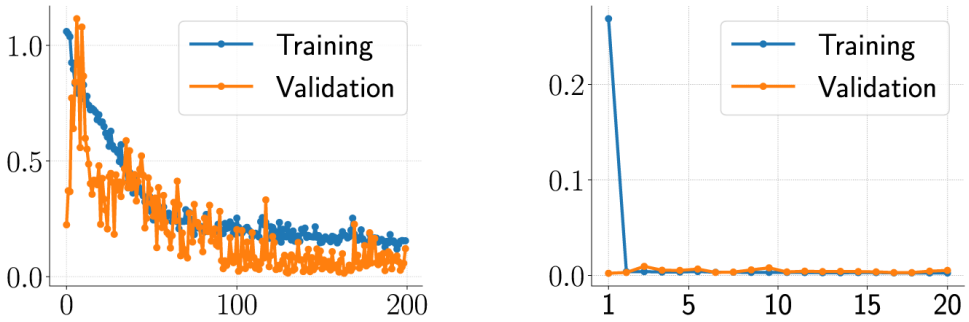
where $G_{y\tilde{y}_{nl}}(f)$ is the cross-spectral density between y and $\tilde{y}_{nl}$, and $G_{yy}(f)$, $G_{\tilde{y}_{nl}\tilde{y}_{nl}}(f)$ are the auto spectral density of y and $\tilde{y}_{nl}$ respectively at a frequency f. The coherence value at frequency f ranges between 0 and 1, where 0 indicates no correlation between the signals at that frequency, and 1 indicates perfect correlation. In the next sections we will be using the MSC to compute the percentage of frequency bins that have coherence greater than a threshold.

UHF-RFID

In this experiment, a UHF-RFID reader, with the antenna pointing towards the ceiling, has been mounted on a element rotating around the axis of the antenna; an RFID passive tag has been mounted on the ceiling in correspondence with the antenna. The UHF-RFID reader is able to measure the phase shift ϕ in the RFID signal backscattered by the RFID tag, which is the measurement that we want to generate through the presented network. The number of inputs n is 5, comprising the antenna rotation angle θ, the position (x_{Tr}, y_{Tr}, z_{Tr}) of an RFID passive tag mounted on the ceiling of the environment and the phase offset ϕ_o depending on the hardware. The number of time-steps for each input to the CRNN network N_s is 25, whilst the number of time-steps for the input to the DAE network M_s is 300.

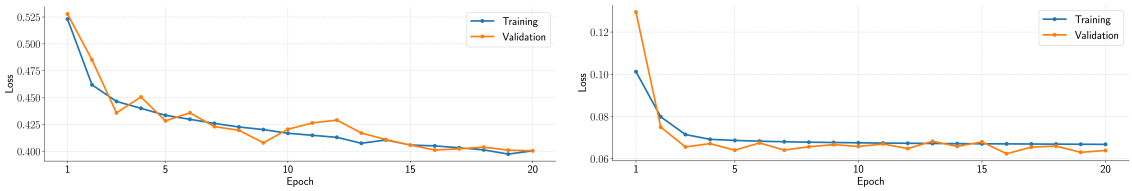
Training Results

The training data obtained from the acquired sensors consist of multiple acquisitions (> 30) of about 1 min long time-series for the n inputs previously defined and the phase shift ϕ in the RFID signal backscattered by the RFID passive tag as the output. These data have been used to train the CRNN on an Intel Core i7-12700H, 2.3GHz, 14 core and 40GB RAM with an NVIDIA GeForce RTX 3050 Ti GPU. The training time is around 28 min. The training and validation loss for 200 epochs of the CRNN are depicted in Figure 3.25a. The model is not



(a) **(b)**

Fig. 3.25 Training and validation loss for (a) UHF-RFID and (b) UWB for the CRNN model.



(a) **(b)**

Fig. 3.26 Training and validation loss for (a) UHF-RFID and (b) UWB for the DAE model.

overfitting for the validation loss is decreasing and the gap between training and validation accuracies is small. Once the CRNN has been trained, it is used to generate the X_i inputs in order to train the DAE model. The training time, in this phase, is around 22 min. The training and validation loss for 20 epochs of the DAE model are reported in Figure 3.26a. Even in this case, the model is not overfitting because the validation loss is decreasing and the gap between the training and validation accuracies is small.

Validation Results

The trained models have been tested with real measurements of $\theta, x_{Tr}, y_{Tr}, z_{Tr}, \phi_o$, as inputs; the output of the network is a deep-generated phase ϕ that closely resembles the real-world data, as shown in Figure 3.27 for one UHF-RFID measurement. The phase ϕ has periodicity of π and it has been unwrapped in the graphs. In order to show the network ability to capture the spectral characteristics, we computed the magnitude-squared coherence between the real and generated measurements at different frequencies. The results are reported in Figure 3.28 for the reference measurement shown in Figure 3.27. In order to have a comprehensive validation of the proposed methodology, we used the network to generate 20 UHF-RFID measurements and computed the average for both the MSE and MSC metrics. Regarding the MSC, we computed the percentage of the generated measurements showing

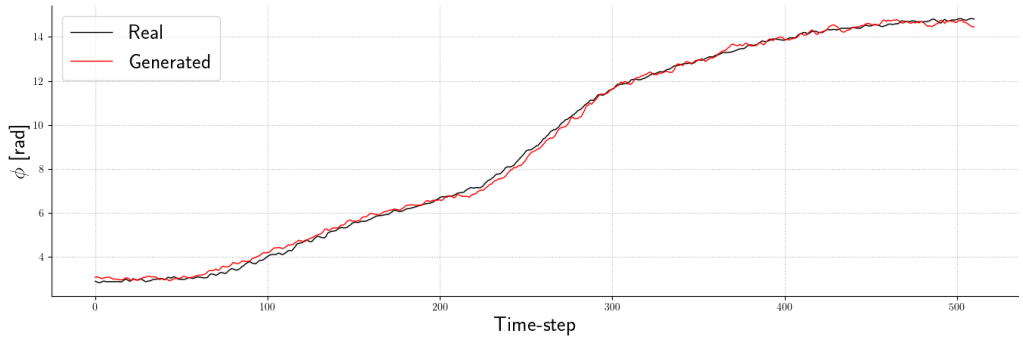


Fig. 3.27 Real and deep generated phase measurements for UHF-RFID. The real measurements are depicted in black, while the generated measurements are depicted in red.

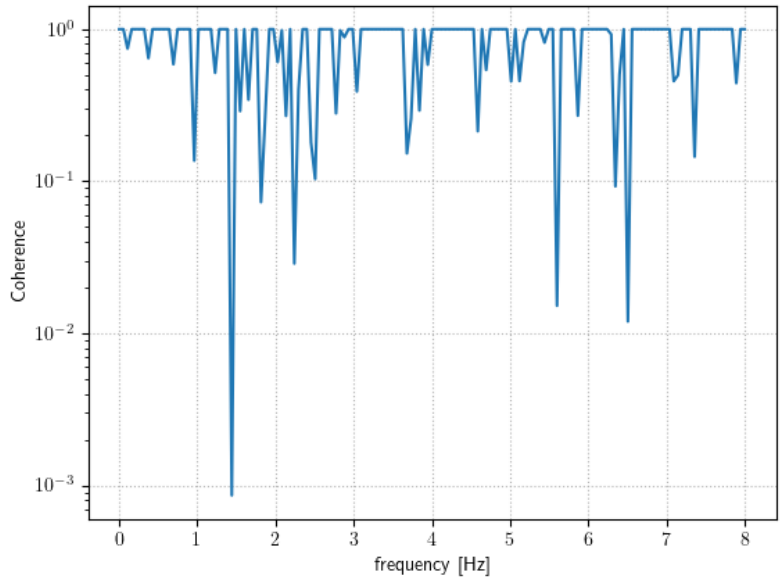


Fig. 3.28 Semi-logarithmic plot of the magnitude-squared coherence between the real and generated phase measurements for UHF-RFID.

a magnitude-squared coherence value greater than a threshold. The chosen thresholds are $(0.5, 0.6, 0.7, 0.8, 0.9)$, from the lowest to the highest coherence value, and are reported in Table 3.4. From this analysis, we can prove the validity of the presented method that shows a small average MSE between the real and generated measurements, that gives us the evidence of how the model is able to produce spatio-temporal measurements that are coherent with the real ones. On the other hand, a high percentage of deep generated measurements (>70 %) present a high level of coherence (> 0.9), proving that the system is also able to capture the spectral characteristics with a high fidelity.

Table 3.4 Average for the mean squared error and percentage of measurements presenting different thresholds of coherence for 20 UHF-RFID deep-generated measurements.

Metrics	Value
MSE	0.028 rad
MSC (> 0.5)	82.12 %
MSC (> 0.6)	78.81 %
MSC (> 0.7)	77.48 %
MSC (> 0.8)	76.82 %
MSC (> 0.9)	74.83 %

UWB

In this experiment, a UWB antenna has been mounted on a unicycle-like robot moving in a cluttered indoor environment (10 m $\times$ 6 m), where UWB tags have been placed in known positions. The UWB antenna is able to read ranges from each UWB tag. The number of inputs n is 6, comprising the robot position (x_r, y_r, z_r) and the position of the UWB tag located in known position (x_{Tu}, y_{Tu}, z_{Tu}). The number of time-steps for each input to the CRNN network N_s is 25, whilst the number of time-steps for the input to the DAE network M_s is 400.

Training Results

The training data obtained from the acquired sensors consist of multiple acquisitions (> 50) of about 5 min long time-series for the previously defined n inputs and the range measurement ρ from UWB antenna to tag as the output. These data have been used to train the CRNN on the same hardware described in Section 3.4.3. The training time is around 35 min. The training loss for 20 epochs of the CRNN is depicted in Fig. 3.25b. The model is not overfitting as the validation loss is decreasing and the gap between training and validation accuracies is small. Once the CRNN has been trained, it is used to generate the X_i inputs in order to train the DAE model. The training time, in this phase, is around 29 min. The training and validation loss for 20 epochs of the DAE model are reported in Figure 3.26b. In this case, also, the model is not overfitting because the validation loss is decreasing and the gap between the training and validation accuracies is small.

Validation Results

The trained models have been tested with real measurements of $(x_r, y_r, z_r, x_{Tu}, y_{Tu}, z_{Tu})$, as inputs; the output of the network is a deep-generated range ρ that closely resembles the

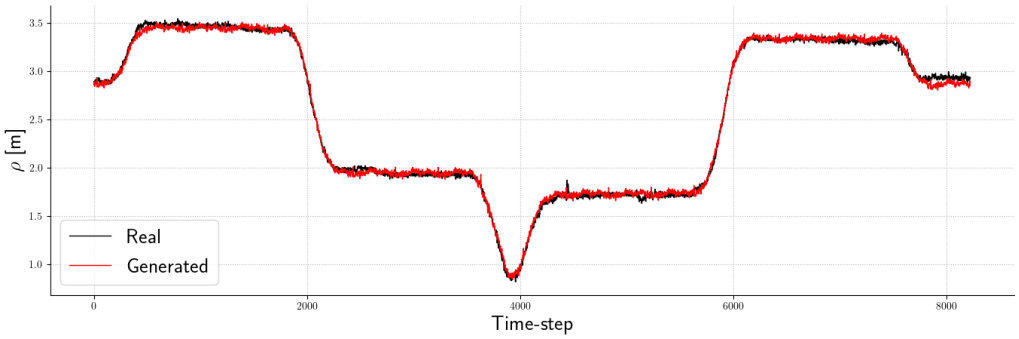


Fig. 3.29 Real and deep generated range measurements for UWB. The real measurements are depicted in black, while the generated measurements are depicted in red.

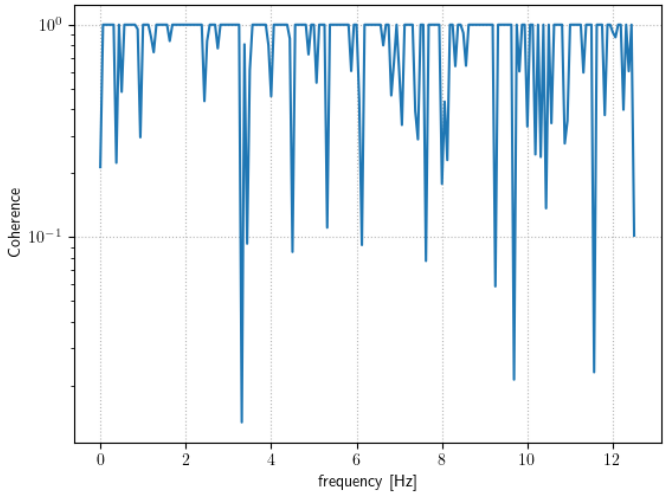


Fig. 3.30 Semi-logarithmic plot of the magnitude-squared coherence between the real and generated range measurements for UWB.

real-world data, as shown in Figure 3.29 for one UWB measurement. In order to show the network ability to capture the spectral characteristics, we computed the magnitude-squared coherence between the real and generated measurements at different frequencies. The results are reported in Figure 3.30 for the reference measurement shown in Figure 3.29. In order to have a comprehensive validation of the proposed methodology, we used the network to generate 30 UWB range measurements and computed the average for both the MSE and MSC metrics. Regarding the MSC, we computed the percentage of the generated measurements showing a magnitude-squared coherence value greater than a threshold as in Section 3.4.3 as reported in Table 3.5. As per the UHF-RFID case, we can prove the validity of the presented method that shows a small average MSE between the real and generated range measurements, that gives us the evidence of how the model is able to produce spatio-temporal measurements that are coherent with the real ones. On the other hand, a high percentage of deep generated

Table 3.5 Average for the mean squared error and percentage of measurements presenting different thresholds of coherence for 30 UWB deep-generated measurements.

Metrics	Value
MSE	0.0029 m
MSC (> 0.5)	83.58 %
MSC (> 0.6)	82.09 %
MSC (> 0.7)	78.60 %
MSC (> 0.8)	76.61 %
MSC (> 0.9)	73.13 %

measurements (>70 %) present a high level of coherence (> 0.9), proving that the system is also able to capture the spectral characteristics with a high fidelity.

3.5 Summary

In this chapter we presented the overview of the methodology for the multi-sensor fusion for resilient robot perception. First in Section 3.1, we introduced the basics to characterize the autonomous systems that have been used to describe the methodology: the agent, the robot and the sensor configuration. For the sensor configuration, we divided the dissertation between proprioceptive and exteroceptive sensors, emphasizing the differences between the two categories and presenting the sensors that have been used in the experiments (that will be presented in Chapter 5). Then, we described the resilient perception subsystem in Section 3.2, highlighting two specific problems of robotic perception: localization and SLAM. For the localization we addressed and presented the problem of global resilient localization, whilst for the SLAM, we have set the methodology for a resilient SLAM using UHF-RFID tags, using TriLateration UHF-RFID tags and using UWB for a unicycle-like robot. Furthermore, we have set the methodology for a resilient SLAM using UWB and Visual Odometry for an autonomous agent.

Subsequently in Section 3.3, we presented different multi-sensor fusion architectures to cope with Odometry-RFID sensor fusion, Visual-UWB sensor fusion and multiple vision sensor fusion. Finally, in Section 3.4 we presented an overview of traditional sensor data acquisition methodologies, also focusing on sensor measurements filtering, and sensor data deep generation. In this last section we presented an approach that involves training deep learning models to learn the underlying patterns and relationships between different types of data. By combining these different sources of information, it becomes possible to generate more accurate and realistic representations of the real world. The process of sensor data deep

generation, presented in that section, has been obtained exploiting Convolutional Recurrent Neural Networks (CRNNs) and, with that, we generated UWB and UHF-RFID sensor data that resemble real world sensor data. As final development, we also presented a new deep neural network model as a combination of CRNN and DAE. This network is able to perform even better than the CRNN alone, characterizing the noise spectrum and making it able to perform a realistic deep generation of sensor measurements.

Chapter 4

Multi-sensor fusion for resilient robot perception

Robot perception is a crucial aspect of robotic systems, allowing robots to understand their environment and interact with it effectively. However, in many real-world scenarios, the perception of a single sensor can be limited, unreliable, or even corrupted. To overcome these challenges, multi-sensor fusion has become an increasingly popular approach to improve the accuracy and robustness of robot perception. By combining data from multiple sensors, robots can gain a more complete and accurate understanding of their environment, enabling them to navigate, interact, and perform complex tasks with greater reliability and resilience. In this chapter, we will explore the concept of multi-sensor fusion and its applications in resilient robot perception. Specifically, we will introduce new techniques to achieve resilient multi-sensor fusion to odometry, visual odometry and range sensors (RFID and UWB). In Chapter 3 we discussed the different types of sensors commonly used in robotics and the challenges associated with fusing their data, giving the theoretical basis for the multi-sensor fusion techniques based on resilience that will be presented in this chapter. The results and application of the presented techniques will be discussed in Chapter 5.

4.1 Odometry-RFID resilient multi-sensor fusion

In this section a resilient multi-sensor fusion system will be presented. In particular, we focused on a SLAM problem using UHF-RFID tags for a unicycle-like robot. The system, comprising a unicycle-like robot mounting a UHF-RFID antenna on top of it has been already presented and described in Section Resilient SLAM using UHF-RFID tags for a unicycle-like robot. The multi-sensor fusion architecture, with an overview of the main

steps of the approach has been presented in Section 3.3.1, where we stated that the resilient SLAM problem will be solved through an EKF algorithm which uses the range and bearing estimation of each detected UHF-RFID tag provided by a set of MHEKF, one for each tag. Here we want to focus on the details about the methodology and algorithm.

4.1.1 Range and bearing estimation

Each tag position with respect to the robot is estimated through a Multi-Hypothesis Extended Kalman Filter (MHEKF) that fuses the phase measurements with the odometry readings as described in [150]. The proposed algorithm, for each tag i, initializes n_M EKF instances $\ell = 1,2,\ldots,n_M$, each one with a different value of the range, since for the periodicity of the RFID phases, several ranges correspond to the same value of a given measured phase. Then, in each EKF instance ℓ, the a priori estimate for range and bearing and the corresponding covariance matrix is computed. The correction step of the EKF algorithm is applied independently in each instance and finally for each instance ℓ two metrics are created and then used to compute its weight. Instances with a too small weight are moved to cycles corresponding to the current phase measurement not covered by other instances. Finally, the estimates $\hat{\rho}_{i,k}$ and $\hat{\beta}_{i,k}$ are taken from the best EKF instance (i.e. by the EKF instance with larger weight). The best EKF instance can be seen as a sensor able to measure the range and the bearing of a specific tag with a particular noise and fault rate associated to it.

Using the best EKF instance ℓ however leads to problems related to the choices of ℓ that can change from one time step to another, given that the weight of the instances change over time. One problem is that these switches between instances strongly affect range and bearing estimations as shown in Figure 4.1. This can introduce big differences between consecutive measurements and a method to cope with this problem, and with the problem related to the frequent switches between instances with similar weight, must be designed. This will be considered in Section 4.1.3.

The overall algorithm, for a single tag i (the i subscript has been omitted), is summarized in the Algorithm 1:

Algorithm 1 Multiple Hypothesis EKF

Initialization. Initialize n_M EKF instances.

Step k: Prediction and Correction. Perform the prediction and correction step. Handle the case of non positive range estimates.

Step k: Weighing Step. Determine the weight w_k^ℓ of each EKF instance $\ell = 1,2,\ldots,n_M$.

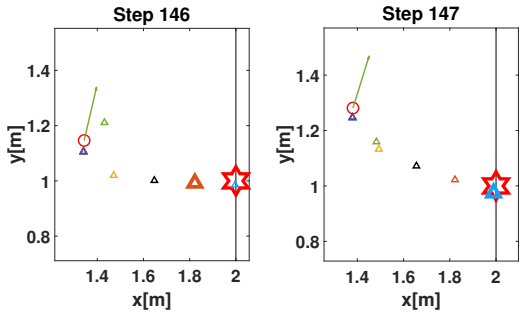


Fig. 4.1 Multi-Hypothesis Extended Kalman Filter execution: triangles represent the estimates produced by the different EKF instances, the selected instances are depicted as bigger triangles and the circle with the segment represents the robot position and heading. A switch between instances is visible from time step 146 (brown instance selected) to time step 147, where the blue instance selected provides an effective estimate of the true tag position, indicated by a red star in $(2,1)$.

Step k: EKF instance cycle transfer. If w_k^ℓ is less than a threshold (e.g. 10^{-13}), set $\hat{\rho}_{k+1}^\ell = -\frac{1}{2K}\phi_{k+1} + \mu\frac{\lambda}{2}$, where μ is a cycle not covered by other instances.
Step k estimate. Take as final estimates $\hat{\rho}_k$ and $\hat{\beta}_k$, the ones provided by the EKF instance with the largest weight.
Set $k = k+1$ and return to 2. □

4.1.2 The Simultaneous Localization And Mapping algorithm.

For each tag i in the environment, the algorithm described in Section Range and bearing estimation is initialized and executed. As an output at each time step k, the estimated ranges and bearings $(\hat{\rho}_{1,k}, \hat{\beta}_{1,k}), \cdots, (\hat{\rho}_{L,k}, \hat{\beta}_{L,k})$ are computed for the L features (i.e. tags). These estimations are then available to perform a simultaneous localization and mapping algorithm with a note: each estimated range and bearing couple can come uniquely from its corresponding tag. This leads to a main simplification for the data association as $(\hat{\rho}_{i,k}, \hat{\beta}_{i,k})$ $(i = 1,2,\cdots,L)$, regarded to as a measurement of feature i, is uncorrelated to any other feature j $(j \neq i)$. The SLAM algorithm adopted in this work uses an Extended Kalman Filter (EKF). The robot pose at time k is $(x_{r,k}, y_{r,k}, \theta_k)$, so the state vector, also including tag positions to map the features in the environment, is:

$$x_k = [x_{r,k}, y_{r,k}, \theta_k, x_{T_1,k}, y_{T_1,k}, \cdots, x_{T_L,k}, y_{T_L,k}]^T$$

The dynamics of the system is:

$$x_{k+1} = \begin{bmatrix} x_{r,k} + u_k \cos(\theta_k) \\ y_{r,k} + u_k \sin(\theta_k) \\ \theta_k + w_k \\ x_{T_1,k} \\ y_{T_1,k} \\ \vdots \\ x_{T_L,k} \\ y_{T_L,k} \end{bmatrix} \tag{4.1}$$

and will be synthetically referred to in this dissertation by

$$x_{k+1} = f(x_k, u_k^e, w_k^e, n_{R,k}, n_{L,k}), \tag{4.2}$$

where u_k^e and w_k^e come from (2.8) and $n_{R,k}$ and $n_{L,k}$ have been already defined before (2.8). The $(3+2L)\times(3+2L)$ Jacobian matrix F_k of the state dynamics with respect to the state is defined as follows:

$$F_k = \left.\frac{\partial f}{\partial x_k}\right|_{x=\hat{x}_k} = \begin{bmatrix} 1 & 0 & -u_k^e \sin(\hat{\theta}_k) & 0 & 0 & 0 \\ 0 & 1 & u_k^e \cos(\hat{\theta}_k) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ & & & & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The $(3+2L)\times 2$ Jacobian matrix with respect to the encoder noises $n_{R,k}$ and $n_{L,k}$ is given by:

$$W_k = \left.\frac{\partial f}{\partial u_k}\right|_{x=\hat{x}_k} = \begin{bmatrix} 0.5\cos(\hat{\theta}_k) & 0.5\cos(\hat{\theta}_k) \\ 0.5\sin(\hat{\theta}_k) & 0.5\sin(\hat{\theta}_k) \\ \frac{1}{d} & -\frac{1}{d} \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \tag{4.4}$$

When a new range and bearing measurement with the robot in state $(x_{r,k}, y_{r,k}, \theta_k)$ and the tag in position $(x_{T_i,k}, y_{T_i,k})$ is available, it can be expressed as follows:

$$h(x_k) = \begin{bmatrix} \sqrt{(x_{r,k} - x_{T_i,k})^2 + (y_{r,k} - y_{T_i,k})^2} \\ \theta_k - \text{atan2}(y_{T_i,k} - y_{r,k}, x_{T_i,k} - x_{r,k}) \end{bmatrix} \tag{4.5}$$

When measurements from L tags are available, (4.5) becomes a vector whose $2L$ elements contain the range and bearing from the L tags. The Jacobian H_k of this measurement function with respect to the state is a $2L \times (3+2L)$ matrix given by:

$$H_k = \left.\frac{\partial h}{\partial x_k}\right|_{x=\hat{x}_k^-} = \begin{bmatrix} \frac{a_1}{d_1} & \frac{b_1}{d_1} & 0 & -\frac{a_1}{d_1} & -\frac{b_1}{d_1} & 0 & \cdots & 0 & 0 \\ \frac{b_1}{d_1^2} & -\frac{a_1}{d_1^2} & 1 & -\frac{b_1}{d_1^2} & \frac{a_1}{d_1^2} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{a_L}{d_L} & \frac{b_L}{d_L} & 0 & 0 & 0 & 0 & \cdots & -\frac{a_L}{d_L} & -\frac{b_L}{d_L} \\ \frac{b_L}{d_L^2} & -\frac{a_L}{d_L^2} & 1 & 0 & 0 & 0 & \cdots & -\frac{b_L}{d_L^2} & \frac{a_L}{d_L^2} \end{bmatrix} \tag{4.6}$$

where:

$$\begin{aligned} a_i &= \hat{x}_{r,k}^- - \hat{x}_{T_i,k}^- \\ b_i &= \hat{y}_{r,k}^- - \hat{y}_{T_i,k}^- \\ d_i &= \sqrt{(\hat{x}_{r,k}^- - \hat{x}_{T_i,k}^-)^2 + (\hat{y}_{r,k}^- - \hat{y}_{T_i,k}^-)^2}. \end{aligned}$$

The EKF-based SLAM algorithm can be summarized as in Algorithm 2.

Algorithm 2 The Extended Kalman Filter for Simultaneous Localization And Mapping

---

Initialization. Let $\rho_{T_i,0}, \beta_{T_i,0}$ be the range and bearing measurements for the i^{th} tag at time 0. Assign the initial estimates:

$$\begin{aligned} \hat{x}_{r,0} &= 0 \\ \hat{y}_{r,0} &= 0 \\ \hat{\theta}_0 &= 0 \\ \hat{x}_{T_i,0} &= \rho_{T_i,0} \cos(\hat{\theta}_0 - \beta_{T_i,0}) \\ \hat{y}_{T_i,0} &= \rho_{T_i,0} \sin(\hat{\theta}_0 - \beta_{T_i,0}) \end{aligned}$$

with $i = 1, \cdots, L$.

Let $k = 0$. Initialize the $(3+2L) \times (3+2L)$ covariance matrix P_k of the estimates as follows:

$$P_0 = \begin{bmatrix} 0_{3\times3} & 0_{3\times2} & \cdots & \cdots & 0_{3\times2} \\ 0_{2\times3} & P_{0,T_1} & 0_{2\times2} & \cdots & 0_{2\times2} \\ \vdots & 0_{2\times2} & P_{0,T_2} & \cdots & \vdots \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0_{2\times3} & \cdots & \cdots & \cdots & P_{0,T_L} \end{bmatrix}.$$

The initial state of the robot is exactly known by definition, since the reference frame adopted by the robot is fixed with the origin in its initial position with the x-axis in the robot direction, so the block of P_0 referring to the robot pose is $0_{3\times3}$. Even if the reference frame is fixed with the origin in its initial position, it is possible to generalize and include the real robot location within an external reference frame. For the tag observations P_0 has L blocks 2×2 in the diagonal P_{0,T_i}:

$$P_{0,T_i} = F_{T_i,0} P_{\rho_{T_i,0},\beta_{T_i,0}} F_{T_i,0}^T,$$

where $F_{T_i,0}$ is the Jacobian matrix of $x_{T_i,0}$ and $y_{T_i,0}$ with respect to $\rho_{T_i,0}$ and $\beta_{T_i,0}$:

$$F_{T_i,0} = \begin{bmatrix} \cos(\beta_{T_i,0}) & -\rho_{T_i,0}\sin(\beta_{T_i,0}) \\ -\sin(\beta_{T_i,0}) & -\rho_{T_i,0}\cos(\beta_{T_i,0}) \end{bmatrix},$$

and $P_{\rho_{T_i,0},\beta_{T_i,0}}$ is the 2×2 covariance matrix related to the EKF selected instance in the MHEKF for the T_i tag at timestep 0.

Prediction step. At each time step k, compute the a priori estimates based on equation (4.1):

$$\begin{aligned} \hat{x}_{r,k+1}^- &= \hat{x}_{r,k} + u_k^e \cos(\hat{\theta}_k) \\ \hat{y}_{r,k+1}^- &= \hat{y}_{r,k} + u_k^e \sin(\hat{\theta}_k) \\ \hat{\theta}_{k+1}^- &= \hat{\theta}_k + w_k^e \\ \hat{x}_{T_i,k+1}^- &= \hat{x}_{T_i,k} \\ \hat{y}_{T_i,k+1}^- &= \hat{y}_{T_i,k} \end{aligned}$$

with $i = 1, \cdots, L$ and where u_k^e and w_k^e depend on the encoder readings. Compute the covariance matrix of the obtained estimates:

$$P_{k+1}^- = F_k P_k F_k^T + W_k Q_k W_k^T, \tag{4.7}$$

where F_k and W_k are, respectively, the Jacobian matrices of the state dynamics with respect to the state (4.3) and to the encoder noise (4.4). The process model covariance matrix Q_k is a diagonal matrix with $K_R|u_{R,k}^e|$ and $K_L|u_{L,k}^e|$ on the diagonal.
Correction step. Let z_{k+1} be the $2L$-dimensional vector of measurements available at time step $k+1$, defined as follows:

$$z_{k+1} = \begin{bmatrix} \hat{\rho}_{T_1,k+1} \\ \hat{\beta}_{T_1,k+1} \\ \vdots \\ \hat{\rho}_{T_L,k+1} \\ \hat{\beta}_{T_L,k+1} \end{bmatrix}.$$

The expected measurement at this stage is $\hat{z}_{k+1}^- = h(\hat{x}_{k+1}^-)$. Hence the correction step of the filter is given by:

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + K_{k+1} n_{k+1},$$

where n_{k+1} is the measurement residual at time $k+1$:

$$n_{k+1} = z_{k+1} - \hat{z}_{k+1}^-, \tag{4.8}$$

and

$$K_{k+1} = P_{k+1}^- H_{k+1}^T (C_{\text{obs},k+1})^{-1} \tag{4.9}$$

is the Kalman gain, H_{k+1} is the Jacobian matrix of the measurements with respect to the state in (4.6) and $C_{\text{obs},k+1}$ is the $2L \times 2L$ covariance matrix associated to the observation vector z_{k+1}, defined as follows:

$$C_{\text{obs},k+1} = H_{k+1} P_{k+1}^- H_{k+1}^T + R_{k+1}, \tag{4.10}$$

where R_{k+1} is the $2L \times 2L$ measurement model covariance matrix constructed following the real noise statistics as:

$$R_{k+1} = \begin{bmatrix} P_{\rho_{T_1,k+1},\beta_{T_1,k+1}} & 0 & \cdots & 0 \\ 0 & \ddots & & 0 \\ \vdots & & & 0 \\ 0 & \cdots & & P_{\rho_{T_L,k+1},\beta_{T_L,k+1}} \end{bmatrix},$$

where $P_{\rho_{T_i,k+1},\beta_{T_i,k+1}}$ is the 2×2 covariance matrix related to the EKF selected instance of the MHEKF for the T_i tag at timestep $k+1$. The covariance matrix is finally updated as follows:

$$P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1}^{-}, \tag{4.11}$$

with I the $(3+2L) \times (3+2L)$ identity matrix. In this dissertation, however we chose the Joseph stabilized version of the covariance correction equation in (4.11) that is more numerically stable because it guarantees the symmetry and positive definiteness of P_{k+1} as long as P_{k+1}^{-} is symmetric and positive definite as suggested in [151]:

$$\begin{aligned} P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1}^{-}(I - K_{k+1}H_{k+1})^T + \\ +K_{k+1}R_{k+1}K_{k+1}^T. \end{aligned} \tag{4.12}$$

□

4.1.3 Dealing with Multi-Hypothesis EKF instance changes.

As already mentioned in Section 4.1.1, the Multi-Hypothesis EKF works with a specific number of EKF instances (10 or 20 are enough to cover the set of all possible cycles), each one initialized on a different cycle corresponding to the initial phase measurement. The instance with larger weight will be chosen to provide the estimates $\hat{\rho}_{i,k}$ and $\hat{\beta}_{i,k}$ (i.e. the i^{th} tag measurements used by the EKF-SLAM algorithm). At each timestep, the choice of the instance could change. When this happens, the range and bearing estimates may be affected by a sudden change. These instance changes are usually caused by an instance stabilization process intrinsically operated by the multiple EKF while selecting the hypothesis which best fits with the measurements. From a SLAM point of view, this behavior can be seen as a condition to initialize a new feature (i.e. another possible tag position in the map) and should be managed with a proper algorithm for partial reinitialization. In order to do so, a methodology to determine if an instance can be considered as stable is proposed in this

section. For a single Multi-Hypothesis EKF, given the $\ell = 1,2,\ldots,n_M$ instances, let $\hat{\ell}$ be the instance with the largest weight at time k, $\hat{\ell}$ is stable at time k if:

$$\hat{i}_j = \hat{\ell}, \forall j \in [k - s_t + 1, k], \tag{4.13}$$

where $\hat{i}_j$ is the instance with the largest weight at time j and s_t is a positive integer to be properly selected.
The instance $\hat{\ell}$ is considered unstable if equation (4.13) is not satisfied for it. This method is applied for all the L Multi-Hypothesis EKFs.

The EKF-SLAM with selective tag state estimation reset

Instability detection step. Perform a check on the stability over the L Multi-Hypothesis EKF instances with the largest weight according to the equation (4.13). If one or more unstable instances are there, then proceed to the next step, otherwise cycle over the instability detection step.
State and covariance update step. If the instance i is marked as unstable at time step k, the state vector and the covariance matrix of the EKF-SLAM must be updated according to the following:

$$\begin{aligned} \hat{x}_{T_i,k} &= \hat{x}_{r,k} + \rho_{T_i,k}\cos(\theta_k - \beta_{T_i,k}) \\ \hat{y}_{T_i,k} &= \hat{y}_{r,k} + \rho_{T_i,k}\sin(\theta_k - \beta_{T_i,k}). \end{aligned} \tag{4.14}$$

The other elements of the state vector are left unchanged as no reset is needed neither in the robot state nor in the tag states which are not affected by instance instability. For the covariance matrix the assumption that $\rho_{T_i,k}$ and $\beta_{T_i,k}$ are considered as real measurements has been made. These measurements have their own covariance matrix taken from the multiple EKF and we can assume that they are independent both from the robot pose and the other tags coordinates. Given the previous assumptions, we can define $\tilde{z} = [\rho_{T_i,k}, \beta_{T_i,k}]^T$ as the range and bearing estimation vector for the T_i tag that has to be reinitialized. Furthermore, if $\tilde{x}_k$ is the EKF-SLAM vector without the coordinates of tag T_i and $\tilde{y} = [\hat{x}_{T_i,k}, \hat{y}_{T_i,k}]^T$ is the vector of the estimate of the tag coordinates, we can define the covariance matrix of the extended state vector $[\tilde{x}, \tilde{z}, \tilde{y}]^T$ as follows:

$$P_{ext} = \begin{bmatrix} P_{\tilde{x}\tilde{x}} & 0_{[3+2(L-1)]\times 2} & P_{\tilde{x}\tilde{y}} \\ 0_{2\times[3+2(L-1)]} & P_{\tilde{z}\tilde{z}} & P_{\tilde{z}\tilde{y}} \\ P_{\tilde{y}\tilde{x}} & P_{\tilde{y}\tilde{z}} & P_{\tilde{y}\tilde{y}} \end{bmatrix},$$

where:

- $P_{\tilde{x}\tilde{x}}$, is the $[3+2(L-1)] \times [3+2(L-1)]$ covariance matrix of $\tilde{x}$ obtained by the covariance matrix P of the EKF SLAM algorithm by deleting the two rows and columns referred to the tag which has to be reinitialized,
- $P_{\tilde{x}\tilde{z}} = P_{\tilde{z}\tilde{x}}^T = 0_{[3+2(L-1)]\times 2}$ is the covariance matrix of $\tilde{x}$ and $\tilde{z}$ that is considered null as the measurements are independent from the state,
- $P_{\tilde{x}\tilde{y}} = P_{\tilde{y}\tilde{x}}^T = P_{\tilde{x}\tilde{x}} F_{\tilde{x}}^T$, is the $[3+2(L-1)] \times 2$ covariance matrix of $\tilde{x}$ and $\tilde{y}$,
- $P_{\tilde{y}\tilde{z}} = P_{\tilde{z}\tilde{y}}^T$, is the 2×2 covariance matrix of $\tilde{y}$ and $\tilde{z}$,
- $P_{\tilde{y}\tilde{y}} = F_{\tilde{x}} P_{\tilde{x}\tilde{x}} F_{\tilde{x}}^T + F_{\tilde{z}} P_{\tilde{z}\tilde{z}} F_{\tilde{z}}^T$, is the 2×2 covariance matrix of $\tilde{y}$,
- $P_{\tilde{z}\tilde{z}}$, is the 2×2 covariance matrix of $\tilde{z}$ given by the Multi-Hypothesis EKF,

where $F_{\tilde{x}}$ and $F_{\tilde{z}}$ are the Jacobian matrices of the relations (4.14) with respect to $\tilde{x}$ and $\tilde{z}$, respectively:

$$F_{\tilde{x}} = \begin{bmatrix} 1 & 0 & -\rho_{T_i,k} \sin(\theta_k - \beta_{T_i,k}) & 0 & \dots & 0 \\ 0 & 1 & \rho_{T_i,k} \cos(\theta_k - \beta_{T_i,k}) & 0 & \dots & 0 \end{bmatrix},$$

$$F_{\tilde{z}} = \begin{bmatrix} \cos(\theta_k - \beta_{T_i,k}) & \rho_{T_i,k} \sin(\theta_k - \beta_{T_i,k}) \\ \sin(\theta_k - \beta_{T_i,k}) & -\rho_{T_i,k} \cos(\theta_k - \beta_{T_i,k}) \end{bmatrix}.$$

The EKF-SLAM covariance matrix P, after the reinitialization of tag T_i position estimate, is:

$$P = \begin{bmatrix} P_{\tilde{x}\tilde{x}} & P_{\tilde{x}\tilde{y}} \\ P_{\tilde{y}\tilde{x}} & P_{\tilde{y}\tilde{y}} \end{bmatrix},$$

where the order of rows and columns could be different from the previous formula according to the T_i tag coordinates positions in the EKF-SLAM state vector. The state and covariance update step shall be run for all the instances marked as unstable. □

4.1.4 Resisting the effects of outliers with resilient EKF-SLAM through hypothesis test and robust estimation

The range and bearing estimation through Multi-Hypothesis EKF is affected by outliers that do not fulfill the assumed stochastic model of extended Kalman filter, so this can be a potential problem for parameters estimation as also mentioned in [152]. In order to resist the effects of the outliers, a resilient extended Kalman filter has been designed. The novel

filter includes a module to detect the presence of one or more outliers in the measurements. If outliers are there, the robust estimation is designed using a modified version of IGGIII scheme (Institute of Geodesy and Geophysics III) based on statistic test of the normalized residual. Furthermore, a fault detection algorithm has been implemented in order to exclude potentially faulty sensors (i.e. wrong MHEKF estimates) from the SLAM algorithm.

Global outliers detection. According to Section 4.1.2, the observation is modeled by a Gaussian distribution with mean and covariance $\hat{z}_k^-$ and $C_{\text{obs},k}$, respectively. So the probability density function can be written as:

$$\begin{aligned} P(z_k) &= N(z_k; \hat{z}_k^-, C_{\text{obs},k}) \\ &= \frac{\exp\left(-\frac{1}{2}(z_k - \hat{z}_k^-)^T (C_{\text{obs},k})^{-1}(z_k - \hat{z}_k^-)\right)}{\sqrt{(2\pi)^{2L}|C_{\text{obs},k}|}}, \end{aligned} \tag{4.15}$$

where $|C_{\text{obs},k}|$ is the determinant of $C_{\text{obs},k}$. If some outliers in the observation are there or if the Gaussian distribution of the observation noise is contaminated with some other distributions, the equation (4.15) will no longer hold and this means that some violations to the assumption or modelling errors could exist. Against a potential measurement outlier, a check if the actual observation is compatible with the assumed model has to be done. The null hypothesis $\mathbf{H_0}$ test is:

$\mathbf{H_0}$: No observation z_i, $i = 1, \cdots, 2L$ is affected by outliers.

The equation (4.15) is used as the relevant null distribution which holds under the assumed model and twice the minus exponent in (4.15) represents the relevant test statistic. The test statistic term is the judging index to detect the modelling errors and this is the square of the Mahalanobis distance from observation z_k to its mean $\hat{z}_k^-$ as deduced in [153]:

$$\begin{aligned} \gamma_k &= M_k^2 = \left(\sqrt{(z_k - \hat{z}_k^-)^T (C_{\text{obs},k})^{-1}(z_k - \hat{z}_k^-)}\right)^2 \\ &= (z_k - \hat{z}_k^-)^T (C_{\text{obs},k})^{-1}(z_k - \hat{z}_k^-), \end{aligned} \tag{4.16}$$

where M_k is the Mahalanobis distance at time step k. The distribution of the test statistic under the null hypothesis is decided so, assuming the null hypothesis is true, γ_k should be Chi-square distributed with $2L$ degrees of freedom. In order to design the statistical test a probability threshold α is needed, below which the null hypothesis will be rejected. The

Table 4.1 Hypothesis test

Decision	$\mathbf{H_0}$ true	$\mathbf{H_0}$ false
Accept $\mathbf{H_0}$	correct	Type II error
Reject $\mathbf{H_0}$	Type I error	correct

α-quantile χ_α of the Chi-square distribution is predetermined so that:

$$\mathscr{P}(\gamma_k > \chi_\alpha) = \alpha, \tag{4.17}$$

where $\mathscr{P}(\gamma_k > \chi_\alpha)$ is the probability of γ_k being larger than χ_α and this should be small as α. Substituting the actual observation $\tilde{z}_k$, we obtain the actual judging index $\tilde{\gamma}_k$, and if this is larger than the α-quantile, the null hypothesis can be rejected and we are confident that one or more outliers occur in the observation as also described in [154]. The null hypothesis $\mathbf{H_0}$ test, being a statistic test, is always accompanied by the probability errors (Type I and Type II errors) with respect to the significance level and the power of a test as in Table 4.1 as suggested in [155]. These kinds of errors cannot be minimized at the same time: studying how to balance these two types of errors is a good tuning parameter for the filter resiliency.
Kalman gain scaling. The global outliers detection gives a methodology to understand if the model does not conform with the specifications, although it does not find which are the outliers in the measurements. The alternative hypothesis to $\mathbf{H_0}$ is that there is at least one outlier in one known observation [156]:

$\mathbf{H_A^{(i)}}$: The observation z_i for some fixed i is an outlier.

The decision can be based on the value of the normalized measurement residual for the observation i at the time step k:

$$w_{k,i} = \frac{n_{k,i}}{\sqrt{c_{n_i n_i}}},$$

where $n_{k,i}$ is the i^{th} element of the measurement residual as in (4.8), $c_{n_i n_i}$ denotes the i^{th} diagonal element of $C_{\text{obs},k}$ given in (4.10). If $\mathbf{H_0}$ holds true, then $w_{k,i} \sim N(0,1)$ as derived in [156]. If an observation i at timestep k is affected by an outlier, the covariance should be inflated and this can be done acting on the Kalman gain K_k. The robust gain matrix factor of Kalman filter $\tilde{K}_{ji}$ (where the time step k has been omitted to make clearer notations from

now on) is:

$$\tilde{K}_{ji} = \begin{cases} K_{ji}, & w_i \leq a_0 \\ K_{ji}\frac{a_0}{w_i}\left(\frac{a_1 - w_i}{a_1 - a_0}\right)^3, & a_0 < w_i \leq a_1 \\ 0, & w_i > a_1, \end{cases} \tag{4.18}$$

where j is the j^{th} element in the state vector and a_0, a_1 are the robust constants of the Kalman gain, usually determined based on the objective requirements. In (4.18) the factor $\frac{a_1 - w_i}{a_1 - a_0}$ is raised to third power, which differs from the implementation of the IGGIII scheme presented in [152] where the same factor is raised to the second power. The reason for this choice is that we want to further decrease the robust Kalman gain factor when the normalized measurement residual is in the range $(a_0, a_1]$. Moreover the measurement vector is composed of range and bearing per each tag, estimated by one Multi-Hypothesis EKF as reported in Section 2.1.4. This leads to the assumption that if the measurement residual related to the range or bearing measurement of the same tag (same multiple EKF) is such that $w_i > a_1$, we set K_{ji} to 0.

Sensor fault detection. The robust gain matrix factor of Kalman adjustments allows a further consideration regarding the measurement reliability associated to a specific sensor providing the couple $(\hat{\rho}_{i,k}, \hat{\beta}_{i,k})$. From this point of view the equation (4.18) can give a further information regarding the number of outliers detected and one can relate its frequency to the reliability of the sensor itself. In particular for the t^{th} tag sensor, one can define the number of times an outlier is detected subsequentially, weighing the outlier whose normalized measurement residual w_i falls within the range $(a_0,\ a_1)$ with a smaller weight compared to the one falling in the range $[a_1,\ \infty)$:

$$\eta_{f,t}(k) = \sum_{n=\bar{k}}^{k} g_n, \tag{4.19}$$

with $\bar{k}$ being the first timestep when the outlier is detected (with $k > \bar{k}$) and:

$$g_n = \begin{cases} 1, & a_0 < w_i < a_1 \\ \bar{g}, & w_i \geq a_1 \\ 0, & \text{otherwise}, \end{cases} \tag{4.20}$$

where $\bar{g}$ is a proper weight with $\bar{g} > 1$. The expression in (4.19) can be used in order to assess how much a sensor is faulty. Based on this fault factor and on the reliability factor defined as

follows:

$$\eta_{r,t}(k) = k - \tilde{k}, \tag{4.21}$$

when no faults are there $\forall k \in [\tilde{k}, k]$ with $\tilde{k}$ the first timestep when the sensor presents no faults, a policy can be designed in order to shutdown the sensor or to restart it when its faults stop increasing. When a tag sensor presents a first fault, the weighted number of faults is computed according to (4.19). When $\eta_{f,t}(k)$ becomes greater than the threshold T_f, that tag sensor t is shutdown. Similarly, after a shutdown when a tag sensor becomes reliable for a proper amount of time T_r (i.e. $\eta_{r,t}(k)$, computed as in (4.21), becomes greater than the threshold T_r), that tag sensor t is restarted. □

4.2 Visual-UWB resilient multi-sensor fusion

In this section a resilient multi-sensor fusion system will be presented for visual odometry and UWB sensors. In particular, we focused on a SLAM problem using UWB antennas for a generic agent moving on an indoor environment. The system, comprising an agent mounting a UWB antenna on top of it has been already presented and described in Section 3.2.2. The multi-sensor fusion architecture, with an overview of the main steps of the approach has been presented in Section 3.3.2, where we stated that the resilient SLAM problem will be solved through an EKF algorithm which uses the range and bearing estimation of each UWB antenna provided by a set of EKF, one for each antenna. Here we want to focus on the details about the methodology and algorithm.

4.2.1 Range and bearing estimation

Each antenna position with respect to the agent is estimated with an EKF fusing range measurements with visual odometry readings. The discretized range and bearing dynamics are given by:

$$\begin{aligned} \rho_{i,k+1} &= \rho_{i,k} - \delta^{vo}_{x,k}\cos(\beta_{i,k}) - \delta^{vo}_{y,k}\sin(\beta_{i,k}) \\ \beta_{i,k+1} &= \beta_{i,k} + \delta^{vo}_{\theta,k} + \frac{\delta^{vo}_{y,k}\cos(\beta_{i,k}) - \delta^{vo}_{x,k}\sin(\beta_{i,k})}{\rho_{i,k}}. \end{aligned} \tag{4.22}$$

Using these equations and the antenna range measurement i, a two dimensional EKF (initialized with the available range measurement and an arbitrary bearing, e.g. 0, characterized by a large standard deviation, e.g. $\pi/3$) can be designed to produce an estimate $(\hat{\rho}_{i,k}, \hat{\beta}_{i,k})$. This estimate will be regarded, from now, on as a sensor able to provide a range and bearing measurement with an uncertainty captured by the covariance matrix of the EKF associated

with the antenna i. However, in the first steps of execution, the EKF reconstructs values far from the real antenna position, so a method to cope with this problem must be adopted.

4.2.2 Simultaneous Localization And Mapping

For each antenna in the environment, the EKF based algorithm in Section 4.2.1 is initialized and executed. At each time step k, the estimated ranges and bearings $(\hat{\rho}_{1,k}, \hat{\beta}_{1,k}), \cdots, (\hat{\rho}_{L,k}, \hat{\beta}_{L,k})$ are computed for the L antennas. These estimations are then available to perform the EKF SLAM algorithm. The state vector, including landmark positions to map the features in the environment, is $x_k = [x_{a,k}, y_{a,k}, \theta_k, x_{u_1,k}, y_{u_1,k}, \cdots, x_{u_L,k}, y_{u_L,k}]^T$. The dynamics of the system is the following:

$$\begin{aligned} x_{k+1} = \Big[& x_{a,k} + \delta^{vo}_{x,k}, y_{a,k} + \delta^{vo}_{y,k}, \\ & \theta_k + \delta^{vo}_{\theta,k}, x_{u_1,k}, y_{u_1,k}, \cdots, x_{u_L,k}, y_{u_L,k} \Big]^T \end{aligned} \tag{4.23}$$

and will be synthetically referred to in this dissertation by:

$$x_{k+1} = f(x_k, \delta^{vo}_{x,k}, \delta^{vo}_{y,k}, \delta^{vo}_{\theta,k}), \tag{4.24}$$

The range and bearing measurements between the agent with pose $(x_{a,k}, y_{a,k}, \theta_k)$ and the antenna in position $(x_{u_i,k}, y_{u_i,k})$ can be defined as follows:

$$h_i(x_k) = \begin{bmatrix} \sqrt{(x_{a,k} - x_{u_i,k})^2 + (y_{a,k} - y_{u_i,k})^2} \\ \theta_k - \text{atan2}(y_{u_i,k} - y_{a,k}, x_{u_i,k} - x_{a,k}) \end{bmatrix} \tag{4.25}$$

When measurements from L antennas are available, (4.5) becomes a vector $h(x_k)$ whose $2L$ elements contain the range and bearing from the L antennas. The EKF-based SLAM algorithm can be easily obtained in a way similar to the one reported in [8] for the case of encoder readings. However, the process model covariance matrix Q_k in the case of Visual Odometry can be created based on information provided on the current pose, assuming that the global features in the environment are only a function of the current pose, as suggested in [157]. Let consider the following measurement model:

$$p^C_f = R^G_C (p^G_f - p^G), \tag{4.26}$$

where $p^C_f = [p_{x,f}\, p_{y,f}\, p_{z,f}]^T$ represents the pose of an ORB-SLAM2 feature with respect to the camera frame, R^G_C the rotation from the camera to the global frame, p^G_f the feature pose with respect to global frame and p^G the camera pose with respect to global frame. In order to

compute Q_k, the following summation can be performed over the n_f features:

$$Q_k = \left(\sum_{f=1}^{n_f} (H_{1,f} H_{2,f})^T \Lambda^{-1} H_{1,f} H_{2,f} \right)^{-1}, \tag{4.27}$$

where $\Lambda = \text{diag}(1/f_x^2, 1/f_y^2)$ (f_x and f_y being the focal lengths in the x and y directions, respectively) and:

$$H_{1,f} = \begin{bmatrix} \frac{1}{p_{z,f}} & 0 & -\frac{p_{x,f}}{p_{z,f}^2} \\ 0 & \frac{1}{p_{z,f}} & -\frac{p_{y,f}}{p_{z,f}^2} \end{bmatrix}$$
$$H_{2,f} = \begin{bmatrix} \lfloor p_f^C \times \rfloor & -R_C^G \end{bmatrix}.$$

4.2.3 Countering the impacts of the outliers

Outliers that do not fit the Extended Kalman Filter's presumed stochastic model can have an impact on range and bearing reconstruction, which could pose a problem when estimating the parameters. A resilient EKF should be created to withstand the effects of the outliers. A module to identify the existence of one or more outliers in the measurements should be part of the innovative filter. If there are outliers, robust estimation is created based on the statistical test of the normalized residual using a modified version of the IGGIII technique described in [152].

4.2.4 Resilient engine for EKF-SLAM

Detection of the global outliers. The observations are shaped as a Gaussian distribution $\mathcal{N}(\hat{z}_k^-, C_{\text{obs},k})$, according to the EKF-SLAM algorithm presented in Section 4.2.2. The presence of observation outliers leads to the possibility that the assumption could be violated; in order to understand if a an outlier for the measurement is there or not, the system must be provided with a test to guarantee that the actual measurement is comparable to that of the alleged model. A test can be defined in order to check that no measurement z_i, $i \in \{1, \ldots, 2L\}$ is influenced by outliers; this test is called the null hypothesis. Taking into account the measurement probability density function, the statistic term can be exploited in order to check for errors in the model. In particular, as provided in [153], the distance of Mahalanobis, denoted with M_k, from measurement z_k and $\hat{z}_k^-$ (being the corresponding mean) can be used as an index to determine possible errors in the model. Specifically, the square of this quantity can be taken into account:

$$M_k^2 = (z_k - \hat{z}_k^-)^T (C_{\text{obs},k})^{-1} (z_k - \hat{z}_k^-). \tag{4.28}$$

When the null hypothesis is true, M_k^2 should follow a Chi-square distribution with a number of degrees of freedom equal to $2L$. A new statistical test must be then defined and, with that, an α threshold below which the null hypothesis must be rejected. Then the Chi-square distribution α-quantile χ_α is defined in such a way that:

$$\mathscr{P}(M_k^2 > \chi_\alpha) = \alpha, \tag{4.29}$$

where $\mathscr{P}(M_k^2 > \chi_\alpha)$ represents the probability that M_k^2 is larger than χ_α. This probability should be small as α. For the current measurement $\tilde{z}_k$, a new index can be defined and denoted with $\tilde{M}_k^2$; when this index results being larger than the α-quantile, the null hypothesis shall be rejected and there is the possibility of having multiple outliers in the current measurement (see [154] for reference).

Scaling for Kalman gain. Although it does not identify which measurements contain outliers, the detection of the global outliers provides a way for determining if the model complies with the parameters. An alternative hypothesis to the null hypothesis can be defined, stating that the measurement z_i is an outlier for multiple fixed i. The value of the normalized observation residual for the measurement i at time step k, is defined as:

$$w_{k,i} = \frac{|n_{k,i}|}{\sqrt{c_{n_i n_i}}}, \tag{4.30}$$

where $n_{k,i}$ is the i^{th} element of the residual of the measurement and $c_{n_i n_i}$ is the i^{th} element on the diagonal of $C_{\text{obs},k}$. The quantity defined in (4.30) can be used to decide whether an observation is an outlier or not. In the case of an outlier, the covariance must be inflated; that can be obtained shaping appropriately the Kalman gain K_k. Hence, a new Kalman gain can be defined as a robust gain matrix factor $\tilde{K}_{ji}$:

$$\tilde{K}_{ji} = \begin{cases} K_{ji}, & w_i \leq a_0 \\ K_{ji}\frac{a_0}{w_i}\left(\frac{a_1 - w_i}{a_1 - a_0}\right)^3, & a_0 < w_i \leq a_1 \\ 0, & w_i > a_1, \end{cases} \tag{4.31}$$

where a_0, a_1 are the robust Kalman gain constants and j is the j^{th} element in the state vector; the time step k has been omitted in (4.31) and it should be noted that the factor $\frac{a_1 - w_i}{a_1 - a_0}$ is raised to the third power as presented in [3]. The rationale for this decision is that when the normalized measurement residual falls inside the range of $(a_0, a_1]$, we wish to further reduce the robust Kalman gain factor.

4.2.5 VO failures: the auxiliary EKF

When the Visual Odometry is not available, the information on the robot motion can be obtained from the available UWB applying a two dimensional Extended Kalman Filter, denoted in the following by *auxiliary EKF*. We have to mention that, during the experiments, when the VO is not available but enough UWB can be seen by the robot, the system is still able to perform the SLAM without any failure.

Let k_0 be the time step when the VO becomes not available. When this occurs, the UWB position estimate is frozen at its current value and only the estimate of the agent position $(x_{a,k}, y_{a,k})$ is updated according to the UWB range measurements. This is performed by applying an Extended Kalman Filter, estimating $(x_{a,k}, y_{a,k})$ only using the available UWB range measurements $\rho_{i,k}$. The filter prediction step of the filter, as the odometry is not available, is applied to the following approximate robot motion model:

$$x_{a,k+1} = x_{a,k} + n_{x,k} \tag{4.32}$$

$$y_{a,k+1} = y_{a,k} + n_{y,k}, \tag{4.33}$$

where the noise terms $n_{x,k}$ and $n_{y,k}$ model the (unknown) agent displacement at step k and are assumed 0 mean Gaussian random variables with standard deviation σ_d depending on the maximum expected agent displacement over a sampling step.

The correction step of the EKF accounts for the range measurements coming from any available UWB i, which position is given by the last estimate $(\hat{x}_{u_i,k_0}, \hat{y}_{u_i,k_0})$ provided by the EKF-SLAM algorithm before the VO failure (i.e. at the switching time k_0).

The initialization of the EKF (i.e. of $(\hat{x}_{a,k}, \hat{y}_{a,k})$ and of the corresponding 2×2 covariance matrix $P_{xy,k}$) is based on the agent position estimate provided by the EKF-SLAM algorithm at the switching time. Specifically, $(\hat{x}_{a,k}, \hat{y}_{a,k})$ is assigned from the latest agent position estimate from the EKF-SLAM, while the covariance matrix $P_{xy,k}$ is initialized with the 2×2 block of the EKF-SLAM covariance matrix related to the agent position estimate.

Then, the equations of the prediction step of the filter are given by:

$$\hat{x}^-_{a,k+1} = \hat{x}_{a,k} \tag{4.34}$$

$$\hat{y}^-_{a,k+1} = \hat{y}_{a,k} \tag{4.35}$$

$$P^-_{xy,k+1} = P_{xy,k} + Q_k, \tag{4.36}$$

where $Q_k = \sigma_d^2 \cdot I_2$, being I_2 the 2×2 identity matrix.

If no UWB measurement is available at step $k+1$, we simply confirm the estimate of the prediction step, by assigning $\hat{x}_{a,k+1} = \hat{x}^-_{a,k+1}$, $\hat{y}_{a,k+1} = \hat{y}^-_{a,k+1}$ and $P_{xy,k+1} = P^-_{xy,k+1}$.

In the case, on the contrary, at step $k+1$ some UWB measurements $\rho_{i,k+1}$ are available, the correction step of the EKF can be applied as follows. Let L_v be the number of UWB measurements available at step $k+1$ and let

$$\hat{\rho}_{i,k+1} = \sqrt{(\hat{x}^-_{a,k+1} - \hat{x}_{u_i,k_0})^2 + (\hat{y}^-_{a,k+1} - \hat{y}_{u_i,k_0})^2} \tag{4.37}$$

be the expected range measurement from UWB i, $i = 1,2,\ldots,L_v$. Let $\hat{\rho}_{k+1}$ be the $L_v \times 1$ vector comprising all the expected available UWB measurements at time step $k+1$ (similarly let ρ_{k+1} be the vector of the corresponding actual measurements). Then, the equations of the correction step of the filter are given by:

$$\begin{bmatrix} \hat{x}_{a,k+1} \\ \hat{y}_{a,k+1} \end{bmatrix} = \begin{bmatrix} \hat{x}^-_{a,k+1} \\ \hat{y}^-_{a,k+1} \end{bmatrix} + K_{k+1}(\rho_{k+1} - \hat{\rho}_{k+1}) \tag{4.38}$$

$$P_{xy,k+1} = (I_2 - K_{k+1} H_{k+1}) P^-_{xy,k+1}, \tag{4.39}$$

where H_{k+1} is the $L_v \times 2$ Jacobian matrix of the measurement model with respect to the state, i.e., for each available UWB measurement i, the i-th raw of H_{k+1} is given by

$$H_{k+1,i} = \frac{1}{\hat{\rho}_{i,k+1}} \left[\hat{x}^-_{a,k+1} - \hat{x}_{u_i,k_0}, \hat{y}^-_{a,k+1} - \hat{y}_{u_i,k_0} \right]. \tag{4.40}$$

The Kalman Gain K_{k+1} is given by

$$K_{k+1} = P^-_{xy,k+1} H^T_{k+1} \\ (H_{k+1} P^-_{xy,k+1} H^T_{k+1} + R_{k+1})^{-1},$$

with R_{k+1} a diagonal $L_v \times L_v$ matrix with element (i,i) given by the variance $\sigma^2_{k+1,i}$ associated with the noise in the measurement coming from the i-th observed UWB at time $k+1$. This noise depends on the error characterizing the UWB sensor and on the uncertainty associated with the estimate of the i-th UWB position. Hence:

$$\sigma^2_{k+1,i} = \sigma^2_\rho + H_{k+1,u_i} P_{u_i} H^T_{k+1,u_i},$$

where:

- σ_ρ is the standard deviation of the measurement error characterizing the UWB sensor;

- P_{u_i} is the 2×2 block of the covariance matrix of the EKF-SLAM related to the estimate $\hat{x}_{u_i,k_0}$ and $\hat{y}_{u_i,k_0}$ of the UWB coordinates at time k_0 (i.e. when the VO starts to become unavailable);

- H_{k+1,u_i} is the derivative of the range measurement from the i-th UWB with respect to the UWB coordinates and this is equivalent, apart from the sign, to the derivative with respect to the agent coordinates, i.e. $H_{k+1,u_i} = -H_{k+1,i}$, where $H_{k+1,i}$ is reported in (4.40).

4.2.6 Switching observer

The switching observer is responsible for the fault detection in the camera sensor which, through the use of the ORB-SLAM2 algorithm, provides an agent pose estimation $\hat{\mathbf{x}}^{vo}$. When the camera or the Visual Odometry algorithm have a fault (e.g. when light conditions change rapidly or some hardware error occurs), the pose estimation is not available and the equations (4.22) and (4.24) cannot be used as the quantities $\delta^{vo}_{x,k}$, $\delta^{vo}_{y,k}$, $\delta^{vo}_{\theta,k}$ are not available. Thus the system needs to switch in order to feed the range/bearing estimation and EKF-SLAM algorithms with a valid agent pose estimation (in our implementation this is provided by the auxiliary EKF presented in Section 4.2.5 and denoted with $\hat{\mathbf{x}}^{uwb}$). Furthermore, when $\hat{\mathbf{x}}^{vo}$ is not available, the EKF-SLAM update step is not performed because the range measurements are already used in the auxiliary EKF algorithm to compute $\hat{\mathbf{x}}^{uwb}$ which is used in the prediction step of the EKF-SLAM. This must be also taken into account by the switching observer as shown in Figure 4.2. The switching law that regulates the way the system acts through the switching signal σ is available and straightforward: when there is a camera fault or the Visual Odometry is not available, the system uses the auxiliary EKF disabling the EKF-SLAM update step, otherwise the Visual Odometry is used and the system works regularly as described in Sections 4.2.1, 4.2.2, 4.2.3. When the VO becomes not available, the last agent position estimate from EKF-SLAM is used by the auxiliary EKF and its covariance matrix is initialized with the EKF-SLAM covariance matrix related to the agent position estimate as previously described in Section 4.2.5. On the other hand, when the VO becomes available, the EKF-SLAM uses the pose and covariance matrix coming from the VO (ORB-SLAM2) as described in Section 4.2.2.

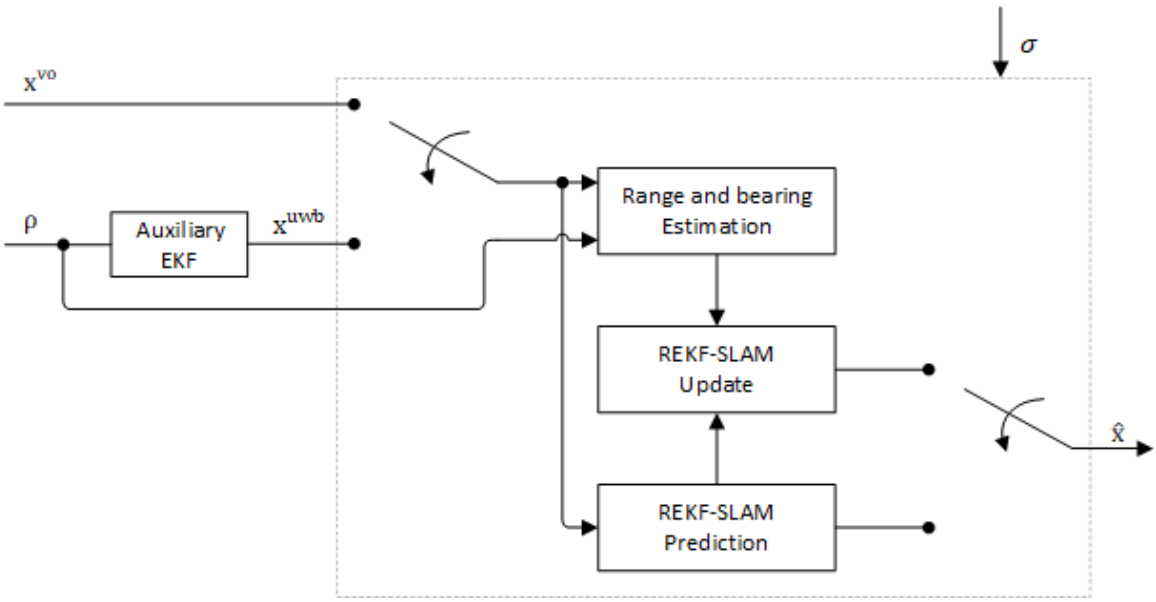


Fig. 4.2 The switching observer schema with the auxiliary EKF, range and bearing estimation, and Resilient EKF-SLAM (REKF-SLAM) blocks.

4.3 Visual-UWB resilient multi-sensor fusion with deep learning approach

In this section a resilient multi-sensor fusion system based on deep learning techniques will be presented for visual odometry and UWB sensors. In particular, we focused on a SLAM problem using UWB antennas for a generic agent moving on an indoor environment. UWB measurements are usually modeled as range measurements affected by Gaussian noise; for more complex analytical models for UWB measurements, they also include biases and other physical effects. However, it is often difficult to model in a precise and effective way the physical effects, especially multi-path, and they are left out of the analytical model. In the presented approach, we resort to a deep neural network, trained on the real UWB measurements, able to incorporate the physical characteristics of those measurements, thus brining more accurate predictable measurements to the update step of the EKF-SLAM algorithm. The presented approach is computationally comparable to the algorithm where the Jacobian matrix can be computed in closed form (due to the availability of real data).

The system comprises an agent mounting a UWB antenna on top of it as presented and described in Section 3.2.2. The multi-sensor fusion architecture, with an overview of the main steps of the approach has been presented in Section 3.3.4, where we stated that the resilient SLAM problem will be solved through an EKF algorithm endowed with a deep learning model for sensor measurements which uses the range and bearing estimation of

each UWB antenna provided by a set of EKF, one for each antenna. Here we want to focus on the details about the methodology and algorithm, especially in the details about the new proposed measurement model achieved with deep learning techniques.

4.3.1 Deep Neural Network architecture

The architecture of the deep neural network is presented in this section. The model must take into account the physical effects and should be able to capture the complexity of the measurement model. In order to design such a model, the deep neural network has been built according to the following components/layers:

- 1 input layer (with 6 input neurons)
- 3 dense layers for each input neuron (the first with 16 neurons and ReLU activation function, the second with 16 neurons and ReLU activation function and the third with 3 neurons and linear activation function)
- 6 dropout layers
- 1 dense layer with 32 neurons and ReLU activation function
- 1 dense layer with 64 neurons and ReLU activation function
- 1 dense layer with 32 neurons and ReLU activation function
- 1 output layer (1 neuron)

The previous architecture design has been achieved through the application of the grid search algorithm to fix the number of neurons on each layer (considered as hyper-parameters of the model). The dropout layers in the middle of the network have been introduced in order to help the network, during the training process, to better generalize from the inputs. The designed deep neural network is reported in 4.3.

Deep Neural Network training with UWB measurements

The deep neural network architecture presented in Section 4.3.1 must be trained with the real sensor measurements acquired in real experiments. In order to do so, we prepared a dataset that has been used in the training phase; the details of the dataset will be presented in Section 5.3.1.

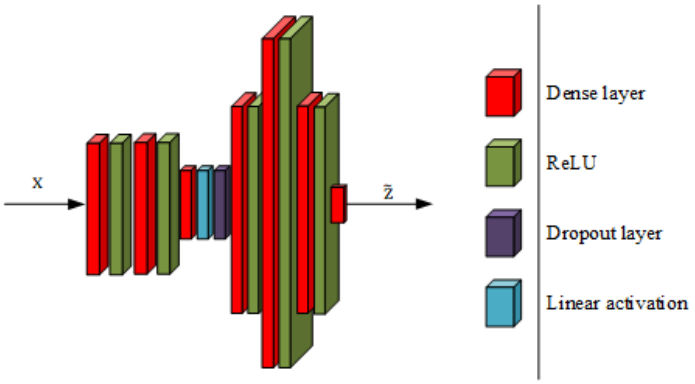


Fig. 4.3 Architecture diagram of the deep neural network for modelling sensor measurements. The model is a combination of dense and dropout layers. The legend on the right reports the layers appearing of the left side of the diagram.

The multi-modal input information is composed of 6 inputs, specifically $[x_{a,k}, y_{a,k}, z_{a,k}, x_{u_i,k}, y_{u_i,k}, z_{u_i,k}]^T$, for each time-step k and for each UWB antenna i. The training has been achieved using the Adam optimizer with a custom loss function defined in Equation 4.41.

$$f_{\text{loss}} = 1 - \frac{\sum_{i=1}^{m}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{m}(y_i - \bar{y})^2 \sum_{i=1}^{m}(\hat{y}_i - \bar{\hat{y}})^2}} + \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2, \tag{4.41}$$

This particular loss function takes into account both the MSE and the Pearson correlation in order to penalize the model for making large prediction errors and to maximize the correlation between the predicted and the real values, as similarly previously defined in Equation 3.46.

4.3.2 Range and bearing estimation

The range and bearing estimation has been achieved resorting to a two dimensional EKF producing an estimate of those quantities. This estimate can be regarded as a sensor able to provide a range and bearing measurements with an uncertainty captured by the covariance matrix of the EKF associated to the UWB antenna i, similarly to that presented in Section 4.2.1.

4.3.3 Simultaneous Localization and Mapping with deep learning

For each antenna in the environment, the EKF based algorithm in Section 4.3.2 is initialized and executed. At each time step k, the estimated ranges and bearings $(\hat{\rho}_{1,k}, \hat{\beta}_{1,k}), \cdots, (\hat{\rho}_{L,k}, \hat{\beta}_{L,k})$ are computed for the L antennas. These estimations are then available to perform the EKF SLAM algorithm. The state vector, including landmark positions to map the

features in the environment, is $x_k = [x_{a,k}, y_{a,k}, z_{a,k}, x_{u_1,k}, y_{u_1,k}, z_{u_1,k}, \cdots, x_{u_L,k}, y_{u_L,k}, z_{u_L,k}]^T$. The dynamics of the system is reported in Equation 4.42.

$$x_{k+1} = \left[x_{a,k} + \delta^{vo}_{x,k}, y_{a,k} + \delta^{vo}_{y,k}, z_{a,k} + \delta^{vo}_{z,k}, x_{u_1,k}, y_{u_1,k}, z_{u_1,k}, \cdots, x_{u_L,k}, y_{u_L,k}, z_{u_L,k}\right]^T \quad (4.42)$$

and will be synthetically referred to in this dissertation by:

$$x_{k+1} = f(x_k, \delta^{vo}_{x,k}, \delta^{vo}_{y,k}, \delta^{vo}_{z,k}), \quad (4.43)$$

The range and bearing measurements between the agent with pose $(x_{a,k}, y_{a,k}, z_{a,k})$ and the antenna in position $(x_{u_i,k}, y_{u_i,k}, z_{u_i,k})$, differently from what has been presented in Section 4.2.2, Equation 4.25 can be computed as follows:

$$h_i(x_k) = h_{DNN}(x_k), \quad (4.44)$$

where h_{DNN} represents the output predicted by the deep neural network presented in Section 4.3.1. When measurements from L antennas are available, (4.5) becomes a vector $h(x_k)$ whose $2L$ elements contain the range and bearing from the L antennas. The EKF-based SLAM algorithm can be easily obtained in a way similar to the one reported in [8] for the case of encoder readings. The process model covariance Q_k is obtained from Equation 4.27.

The measurement model $h_{DNN}(x_k)$ is not in an analytical closed form, thus the Jacobian of the measurement function $H_k = \frac{\partial h}{\partial x_k}\Big|_{x=\hat{x}_k^-}$ (whose analytical form, for a slightly different set of state variables, is reported in Equation 4.6) cannot be expressed analytically, but it has to be computed numerically. In order to do so, we choose a small perturbation value ε that should be small enough to avoid introducing significant errors but large enough to avoid numerical precision issues. Then, for each element of the state vector that the measurement function depends on, we create two slightly perturbed versions of the state vector. One version should be the "actual estimate," and the other should be a value close to the estimate, achieved by adding or subtracting ε from the actual estimate. Then, we evaluate the measurement function at both the actual estimate and the perturbed estimate for each element of the state vector. And we calculate the derivative for each element of the state vector using the central difference formula. Given that, Equation 4.6 can be rewritten as follows:

$$H_k = \frac{\partial h}{\partial x_k}\Big|_{x=\hat{x}_k^-} =$$

$$\begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} & H_{1,4} & H_{1,5} & H_{1,6} & \cdots & 0 & 0 & 0 \\ H_{2,1} & H_{2,2} & H_{2,3} & H_{2,4} & H_{2,5} & H_{2,6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ H_{2L-1,1} & H_{2L-1,2} & H_{2L-1,3} & 0 & 0 & 0 & \cdots & H_{2L-1,3+3L-2} & H_{2L-1,3+3L-1} & H_{2L-1,3+3L} \\ H_{2L,1} & H_{2L,2} & H_{2L,3} & 0 & 0 & 0 & \cdots & H_{2L,3+3L-2} & H_{2L,3+3L-1} & H_{2L,3+3L} \end{bmatrix}$$

$$H_{i,j} = \frac{h_{DNN,i}(\hat{x}_k + \varepsilon e_j) - h_{DNN,i}(\hat{x}_k - \varepsilon e_j)}{2\varepsilon}, \tag{4.45}$$

where $H_{i,j}$ is the derivative of the i-th measurement function element with respect to the j-th state variable, ε is the perturbation value and e_j is a vector with zeros in all elements except the j-th element, which is 1. The value for ε should be small and it has been chosen as 10^{-6}.

The EKF-SLAM problem can then be solved as reported in Section 4.1.2, computing the correction step, Kalman gain and updated covariance matrix, using the Jacobian H_k as defined in Equation 4.45.

4.3.4 Resilient engine for Deep EKF-SLAM

The results related to the resilient engine presented in Section 4.2.4 are still valid also in the case of the measurement model provided by a deep neural network. In particular, the Mahalanobis distance (Equation 4.28) will provide a better index of how the real measurements are far from the predicted measurements, if the DNN model is more precise in capturing the details and characteristics of the sensor measurements compared to the analytical model. At the same time, the Kalman gain scaling, that relies upon the observation residual (i.e. $n_k = z_k - \hat{z}_k^-$), will present better results with the DNN measurement model compared to the analytical model presented in Section 4.2.2 if it is confirmed that the DNN model is more precise than the analytical model. The discussion and comparisons of the two models will be presented in the experiments reported in Section 5.3.1.

4.4 Multiple visual sensor fusion

In this section a multiple visual sensor fusion system will be presented for visual odometry sensors. In particular, the proposed sensor fusion architecture is based upon the hypothesis that there are several visual odometry systems for the agent.

The system has been already presented and described in Section 3.3.3; the details of the approach will be presented in this section.

4.4.1 Pre-filtering

Like other sensors, visual sensors are susceptible to noise, which also has an impact on the precision of the localization systems that analyze their samples. The location measurements' accuracy might be harmed by this noise, which could potentially produce outliers. Samples must thus undergo adequate filtering prior to entering the sensor fusion system. The proposed approach is divided into two steps: first, each measurement is subtracted its previous value (as reported in Section 3.3.3) to obtain a difference sample $\Delta_i[k]$ for $i = 1, \ldots, n$, then, this sample is filtered by the operator $\mathfrak{F}$. By using different samples, one can reduce systematic errors that could negatively impact localization systems. Additionally, using different samples makes it easier to implement subsequent stages because each sample's absolute value is reduced to the bare minimum required to hold useful information, minimizing the impact of quantization or representation errors brought on by microprocessors. Finally, the purpose of the operator $\mathfrak{F}$ is to further smooth the samples and reduce the impact of outliers. In the literature, a similar problem has been solved in the context of image processing by applying median filtering, i.e. averaging under the L_1 norm [158–160]. Thus, the $\mathfrak{F}$ operator is designed as a median filter, a discrete-time nonlinear filter that returns the median over a buffer of samples, whose length can be tuned as a parameter of the sensor fusion system, and implemented using the efficient Quicksort algorithm [161].

4.4.2 Blending

The design of the blending map $\mathfrak{B}$ is limited by the only information available to it as in Section 3.3.3: the position difference samples and the state of each sensor. A meaningful and effective way to combine this limited information is to perform a weighted average of the difference samples. Let $\bar{\Delta}_f[k] = [\Delta_{1,f}[k]', \ldots, \Delta_{n,f}[k]']'$, $\bar{s}[k] = [s_1[k], \ldots, s_n[k]]'$, and $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ such that $\sum_i \alpha_i = 1$. The blending map is given by:

$$\mathfrak{B}(\bar{\Delta}_f[k], \bar{s}[k]) = \sum_{i=1}^{n} \bar{\alpha}_i[k] \Delta_{i,f}[k]\,, \tag{4.46a}$$

$$\bar{\alpha}_i[k] = s_i[k]\alpha_i \left(1 + \frac{\sum_{j=1}^{n}(1 - s_j[k])\alpha_j}{\sum_{j=1}^{n} s_j[k]\alpha_j}\right) \quad i = 1, \ldots, n\,. \tag{4.46b}$$

The structure of (4.46) is simple, but it has many features. First, $(\alpha_1, \ldots, \alpha_n)$ can be tuned to account for different accuracy and overall reliability of each sensor. Additionally, the sensors' condition affects how the samples are fused, enabling the system to withstand momentary or long-term failures of some of its modules. Last but not least, the map (4.46a) is highly effective in terms of implementation since it resolves to a sum of products that can be run

very rapidly on any CPU. The blended difference sample is then combined with the last position sample after being post-processed using a median filter.

Chapter 5

Applications: resilient multi-sensor fusion for autonomous agents. Numerical and experimental results

In recent years, the field of autonomous systems has witnessed remarkable advancements, revolutionizing various industries such as transportation, robotics, and surveillance. Autonomous agents, equipped with sophisticated sensor suites, have become integral components of these systems, enabling them to perceive and interact with the surrounding environment in real-time. However, ensuring robust and resilient perception remains a significant challenge due to the complexity and uncertainty inherent in real-world scenarios.

The key to addressing this challenge lies in effectively integrating information from multiple sensors through the process of multi-sensor fusion. Multi-sensor fusion leverages the complementary strengths of different sensor modalities, mitigating the limitations of individual sensors and enhancing the overall perception capabilities of autonomous agents. By fusing data from diverse sensors, such as cameras, LiDARs, radars, and GPS, autonomous agents can obtain a more comprehensive and accurate understanding of their surroundings. This chapter focuses on the application of resilient multi-sensor fusion for autonomous agents (mobile robots and generic agents), exploring the integration of classical and deep learning techniques to enhance the perception capabilities of these agents. Resilience is a critical aspect, as it ensures that the perception system remains robust and reliable even in the face of sensor failures, adverse weather conditions, or unexpected environmental changes. By designing resilient multi-sensor fusion algorithms, we aim to develop perception systems that can adapt and maintain accurate perception under challenging circumstances.

The chapter is structured as follows: first, we provide the results of an application for a mobile robot system with sensors based on odometry and RFID antenna and tags.

Then we present the results of an application for a generic agent with visual odometry and UWB antennas, moving in an unknown environment. Subsequently, we explore the integration of deep learning techniques into multi-sensor fusion frameworks. Deep learning has revolutionized many domains of artificial intelligence, and its application to multi-sensor fusion holds great promise for enhancing perception capabilities. We discuss the potential benefits of deep learning in multi-sensor fusion and highlight various architectures and methodologies that have been proposed in the literature.

To address the challenges of resilience, we then introduce novel approaches for designing resilient multi-sensor fusion algorithms. We investigate techniques for sensor failure detection to ensure that the perception system can adapt and maintain accurate perception even in the presence of sensor failures or degraded sensor data. We also explore strategies for handling uncertainties and adversarial conditions in the environment, enabling the autonomous agents to make informed decisions. Throughout the chapter, we provide comprehensive evaluations and comparisons of different fusion techniques, both classical and deep learning-based, using real-world datasets and simulation environments. We assess the performance of these techniques in terms of accuracy, robustness, computational efficiency, and adaptability to different environmental conditions.

In summary, this chapter aims to contribute to the growing body of research on multi-sensor fusion for autonomous perception by focusing on the development of resilient algorithms. By combining classical and deep learning techniques, we strive to enhance the perception capabilities of autonomous agents, enabling them to operate in complex and uncertain environments. The findings and insights presented in this chapter will contribute to the advancement of autonomous systems and pave the way for more intelligent and reliable autonomous agents in various applications.

5.1 Application to an odometry-RFID system

In this section the results of the application of the resilient multi sensor fusion paradigm presented in Section 4.1 will be reported. The results will be presented both from several sets of simulations that have been run (in Section 5.1.1) and on real-world experiments that have been conducted and whose details will be presented in Section 5.1.2.

5.1.1 Numerical investigation and examples

This section comprises two sets of simulations, one with synthetic UHF-RFID data corrupted only by a Gaussian noise and the other with UHF-RFID data generated by means of a

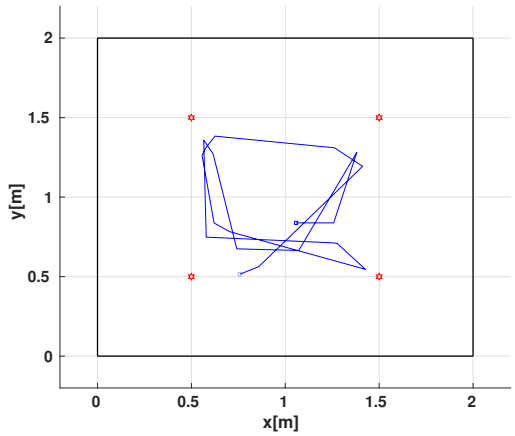


Fig. 5.1 Simulation scenario considered in Section 5.1.1, with the tags in position $(0.5, 0.5)$, $(1.5, 0.5)$, $(0.5, 1.5)$, $(1.5, 1.5)$ and one possible random robot trajectory. The small square box represents the starting point of the considered trajectory, which is 14.63 m long and is covered in 2000 simulation steps. The small circle represents where the robot stops.

numerical calculation based on suitable beam-tracing algorithms [162–164]. The scenario considered in the first case is a 2×2 m^2 indoor environment without multipath effects whilst, in the second case, a 6×6 m^2 room containing some furniture is considered, as detailed below. The robot performs random paths of the type reported in Figure 5.1. These paths are generated by randomly selecting the initial position and orientation of the robot and with the odometry readings numerically generated with a Gaussian error. When going straight, the robot proceeds at constant speed covering about 1cm in each time step. When performing a turn, the robot covers about 5° per time step. The duration T of the simulations considered in this section is 2000 steps, corresponding to an average of about 15 m of traveled distance (depending on the number of turns performed). The parameters of the robot are as follows: $d = 26$ cm, $K_R = K_L = 0.01$ cm. In the two simulation sets the UHF-RFID tags are located at a height of 2.5 m with respect to the robot antenna and the estimated height is perturbed over each simulation with a random error of ± 3 cm (this error is due to the fact that the ceiling height is known with a certain approximation). Moreover the unknown offset ϕ_o on the phase measurements depending on the hardware, as stated in Section 3.1.3, is also considered in the simulations as a random value from 0° to 360° and different in each simulation.

After discussing in Section Parameter tuning some issues related to the choice of the main parameters appearing in the algorithms, a set of simulations is reported in Section Numerical examples to illustrate the behavior of the proposed approach in a noisy environment, where phase measurements are characterized by a Gaussian noise with standard deviation $\sigma_\phi = 10°$. Furthermore, in Section Numerical analysis with multipath affected signals a set of

simulations is reported where the phase measurements are generated by numerical calculation based on beam-tracing algorithms, adding the Gaussian noise, considering a 6×6 m^2 indoor environment with some furniture to also consider the effect of multipath on the system, shown in Figure 5.2.

The numerical model we have used takes into account the materials of the environment and the propagation effects. In particular, the model has been developed for the central frequency of the allowed European band for RFID (i.e. 867 MHz), walls are considered made of concrete having relative permittivity 6 and loss tangent 0.25, while furniture are supposed made of wood with relative permittivity 1.7 and loss tangent 0.001. The wave propagation algorithms account for multiple reflections from walls and furniture and wedge diffraction. Since the interactions (i.e. reflections and diffractions) with the environment may be mitigated by particular radiation properties of the reader's and tag's antenna (e.g. directive antennas may mitigate multipath effects), we use a half-space isotropic model of antennas in order to test the SLAM algorithms in the most possible general conditions. Considering the reader's antenna is near the floor and points toward the ceiling, while the tag's antenna is on the ceiling and points toward the floor, both reader's and tag's antenna are modelled as isotropic sources in the half-space they are pointing. A half-space isotropic source accounts for multipath coming from all directions in that half-space consequently it models the most general multipath condition. Throughout the following sections, the results of the presented methodology have been compared to those of the methodology proposed in [165], showing the overall performance improvements both in terms of SLAM estimates and computational complexity.

Parameter tuning

The proposed algorithm relies on the following parameters to be properly tuned:

- s_t, which is a constant used to decide if the current instance for a tag estimation in the Multi-Hypothesis EKF is stable or not, as number of timesteps;
- α-quantile χ_α in (4.17);
- a_0 and a_1 (with $a_0 < a_1$) constants, appearing in (4.18);
- $\bar{g}$ constant, weighing the outliers whose normalized measurement residual is greater than a_1;
- T_f is the threshold beyond which the sensor is considered faulty;
- T_r is the threshold beyond which the sensor is considered reliable.

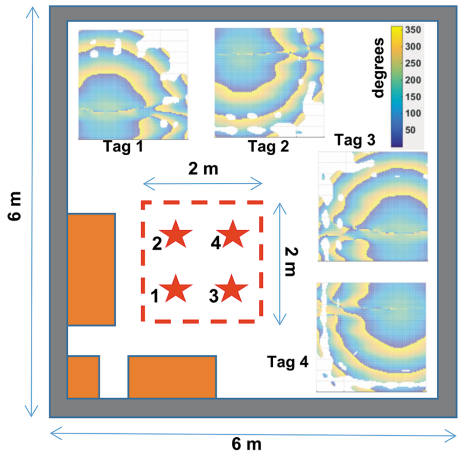


Fig. 5.2 Indoor scenario with four tags located in position $(1.5,1.5)$, $(1.5,2.5)$, $(2.5,1.5)$, $(2.5,2.5)$. The robot moves in the dashed red 2×2 square area, close to wood furniture, which introduce multipath. The 4 insets show the phase of the four tags measured in the 2×2 considered area: multipath effects, which perturb phases and create voids (white spots on the diagrams where the signal is not received), are clearly visible (in ideal conditions we would obtain concentric circles). The vertical bar indicates the phase value in degrees.

As for s_t, a too small value of this parameter (e.g. s_t less than 5 steps) makes the instance stable even if it is switching quite frequently and this should be avoided. On the other hand a high value of s_t (e.g. s_t greater than 40 steps) implies that the instance is seen as stable after a considerable amount of time, and going beyond this value could lead to evaluate the instance as unstable continuously. Based on numerical results, a good trade off has been obtained by taking $s_t = 20$ steps. For the outlier detection, α-quantile χ_α in (4.17) is chosen from the Chi-square distribution table for 8 degrees of freedom, as 4 tags are considered in the scenario and each tag has 2 (for range and bearing measurements). For the 8 degrees of freedom Chi-square distribution with the significance level being 1 %, it is 20.09. The robust Kalman filter gain relies upon the constants a_0 and a_1 (with $a_0 < a_1$), appearing in (4.18). Based on the work in [166], a_0 ranges in $[1.0,2.5]$ and a_1 ranges in $[3.0,8.0]$. For the simulations $a_0 = 1.5$ and $a_1 = 3.5$ have been chosen in order to achieve a good trade-off between marking an observation as an outlier and integrating it as a correct measurement (eventually adjusting it). The sensor fault detection method in Algorithm 4.1.4 relies upon the constant $\bar{g}$ weighing the outliers whose normalized measurement residual is greater than a_1, as previously fixed. The value $\bar{g} = 2$ has been chosen as it allows the algorithm to weigh twice the outliers that zero the Kalman gain elements $\tilde{K}_{ji}$ in respect to the outliers that reduces the Kalman gain elements $\tilde{K}_{ji}$ according to (4.18). Furthermore, T_f and T_r are two important parameters for the fault/reliability detection, T_f being the threshold beyond which the sensor

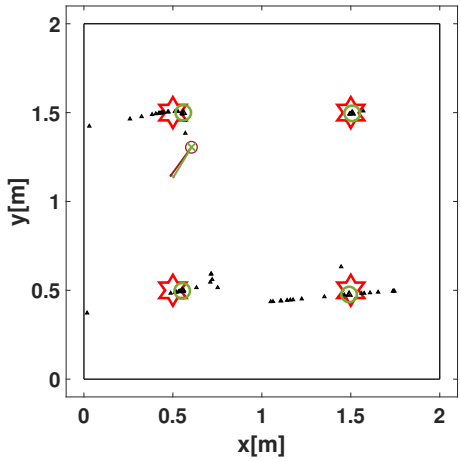


Fig. 5.3 Illustration of the algorithm during the execution: the green circles highlight the Multi-Hypothesis EKF selected instance (i.e. the measurement feeding the EKF-SLAM algorithm), while the small black triangles represent the MHEKF instances. The robot ground truth pose is depicted with a red circle and its orientation with an arrow pointing out of the circle whilst the robot estimated pose is represented by a green x mark and its orientation with an arrow pointing out of the mark. The red stars represent the real tags position.

is considered faulty. If a too small value is chosen for this threshold (e.g. $T_f < 5$), then the system will shutdown the sensors even if they are still working with reduced quality. On the other hand if its value is too high (e.g. $T_f > 20$), the system will only catch few faulty situations. Given that, a good trade off has been obtained by taking $T_f = 10$, which means that the system is capable of shutting down the malfunctioning sensor in the worst case after 10 timesteps. A similar argument applies to parameter T_r which has been set to 8.

Numerical examples

In this section multipath effects are not considered and only a Gaussian noise has been added to the ideal phase measurements. The scenario is the one described in Figure 5.1. The execution of the robust EKF-SLAM algorithm is depicted in Figure 5.3. The path traveled by the wheeled robot is reported in Figure 5.4, where the ground truth trajectory (in blue) is shown together with the trajectory estimated by the robust EKF-SLAM (in red).

The same set of simulations has been run with the SLAM algorithm presented in [165], in order to compare their performances with the indexes proposed in that paper. As for the robot position estimation, we first compute the difference between the true and the estimated distance of the robot from the position of the four tags in the various steps k of the simulation,

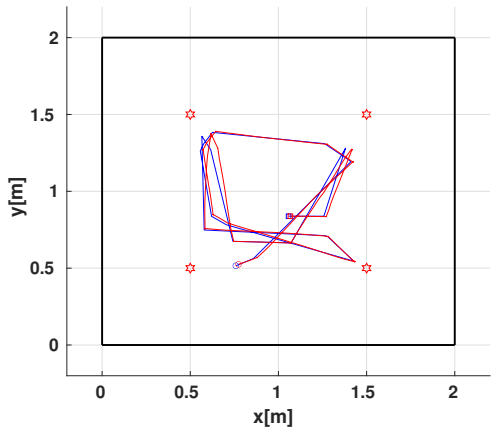


Fig. 5.4 The ground truth robot trajectory (blue) and the trajectory estimated through the EKF-SLAM algorithm (red).

i.e.

$$e_{ri,k} = \left| \sqrt{(x_{r,k} - x_{T_i})^2 + (y_{r,k} - y_{T_i})^2} - \sqrt{(\hat{x}_{r,k} - \hat{x}_{T_i,k})^2 + (\hat{y}_{r,k} - \hat{y}_{T_i,k})^2} \right|.$$

Then, an average robot position estimation error is computed by averaging the previous quantity over the 4 tags and considering the last 1000 steps of the simulation:

$$e_r = \frac{1}{4} \sum_{i=1}^{4} \left[\frac{1}{1000} \sum_{k=T-999}^{T} e_{ri,k} \right],$$

with $T = 2000$ being the duration of each simulation.

As for the tag position estimation, we similarly define the difference between the true and the estimated distance among the four tags in the various steps k of the simulation, i.e.:

$$e_{tij,k} = \left| \sqrt{(x_{T_i} - x_{T_j})^2 + (y_{T_i} - y_{T_j})^2} - \sqrt{(\hat{x}_{T_i,k} - \hat{x}_{T_j,k})^2 + (\hat{y}_{T_i,k} - \hat{y}_{T_j,k})^2} \right|,$$

with $i = 1,2,3$ and $j = i+1,\ldots,4$. Then, an average tag position estimation error is computed by considering the average of the various estimation errors in the last step of the simulation:

$$e_t = \frac{1}{6} \sum_{i=1}^{3} \sum_{j=i+1}^{4} e_{tij,T},$$

where 6 is the total number of distance errors $e_{tij,T}$, $i = 1,2,3$ and $j = i+1,\ldots,4$.

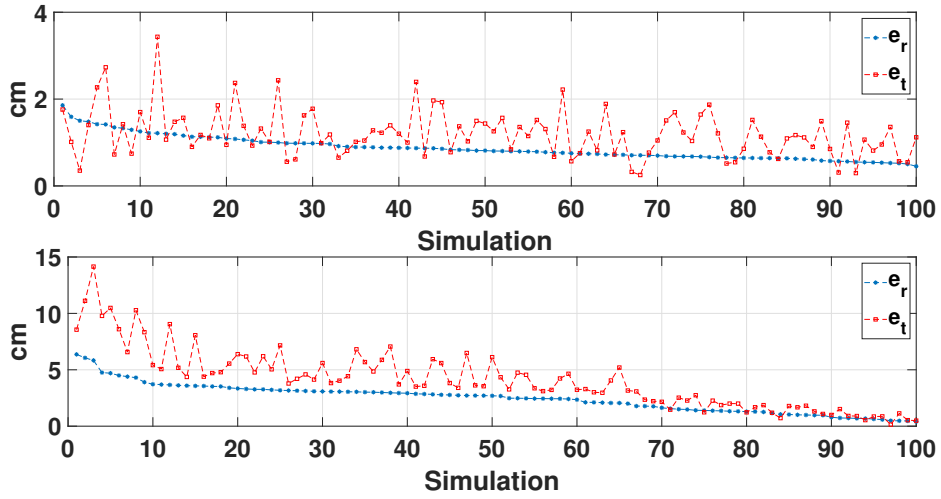


Fig. 5.5 This figure shows the trend of the errors for both robot (e_r) and tags (e_t) position estimation for the EKF-SLAM presented in this discussion (top) and for the particle filter proposed in [165] (bottom).

Table 5.1 Robot and tags position average estimation errors with standard deviations (in cm)

e_r		STD	e_t		STD
Proposed approach	0.869	$\pm$0.27	Proposed approach	1.204	$\pm$0.55
Approach [165]	2.479	$\pm$1.25	Approach [165]	4.110	$\pm$2.64

The results of the simulations for the two algorithms are depicted in Figure 5.5 where the errors e_r and e_t, previously defined, for 100 simulations for the method presented in this discussion (top) and the results for the algorithm proposed in [165] (bottom) are showed. This figure shows how both the errors for the robot and tags position estimation with the presented EKF-SLAM are smaller compared to the SLAM approach in [165]. Table 5.1 shows the estimation errors average over all the simulations for the two methods. A further simulation has been carried out in order to assess the performance of the resilience module; in particular, the tag $(1.5, 1.5)$ has been shifted instantly from its original position to $(0, 1.5)$ from time-step 1000. Over the 100 simulations, the average error e_r is 0.039 m for the presented method and 0.097 m for the approach in [165]; e_t is 0.026 m for the presented method and 0.209 m for the approach in [165]. The tag perturbation heavily affects the SLAM approach in [165] as its average error for tag position estimate for the shifted tag is 0.945 m while is 0.034 m with the presented approach.

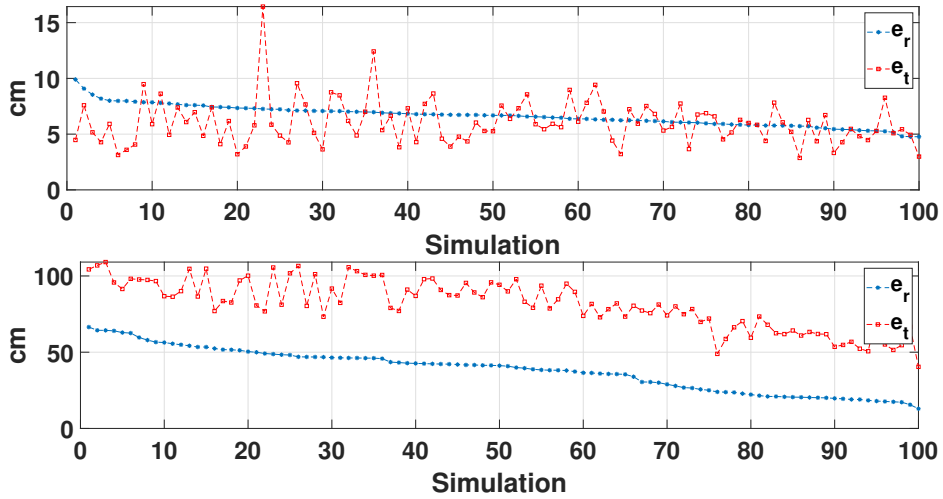


Fig. 5.6 Error trends for both robot (e_r) and tags (e_t) position estimation for the EKF-SLAM (top) and for the particle filter approach (bottom) in the case of multipath affected measurements.

Numerical analysis with multipath affected signals

This section refers to the case described in Figure 5.2, where phase measurements are generated using the ray tracing software mentioned above and are corrupted with a Gaussian noise with standard deviation 10°. 100 simulations have been run with both the approach described in this discussion and the one presented in [165]. Fig. 5.6 shows the results of the simulations for both methodologies. From this figure and from table 5.2 it is clear how the presented EKF-SLAM methodology outperforms the approach presented in [165] both in terms of performance and, also, from a computational point of view. In fact, the simulations ran on a AMD Ryzen 7 3800x 8-core processor 3.9GHz with 32GB RAM on Linux Ubuntu 20.04 with Matlab R2021a, showed an average computation time per simulation of 12.82 s for the presented method and 91.59 s for the method in [165]. The difference of the computation times would have been even more pronounced with more tags, as the approach in [165] needs about 1000 initial instances for any new tag, while the presented method needs a few dozen. For the presented method, the computation time for each time step (taking into account that each simulation consists of 2000 time steps) is 0.0064 s: this low computation time allows an implementation on a real robot even with small computational power with a real-time fashion.

Table 5.2 Robot and tags position avg estimation errors with standard deviations, under multipath effects (in cm)

e_r		STD	e_t		STD
Proposed approach	6.635	$\pm$0.92	Proposed approach	6.016	$\pm$2.03
Approach [165]	38.363	$\pm$13.81	Approach [165]	81.553	$\pm$16.11

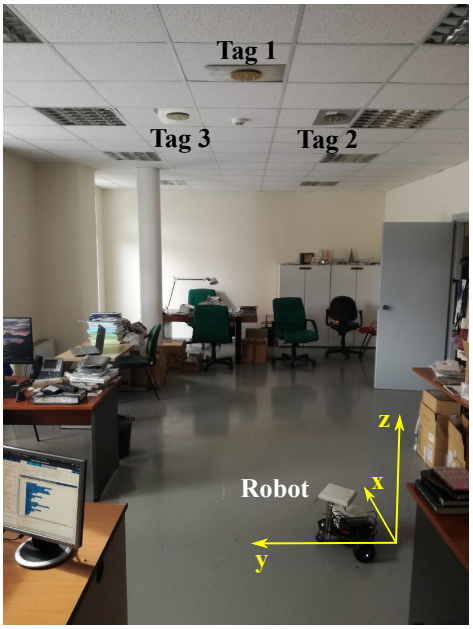


Fig. 5.7 The experimental setup, with three tags on the ceiling, the robot during its mission and the (x,y,z) frame adopted by the robot to solve the SLAM problem, selected according to the initial pose of the robot.

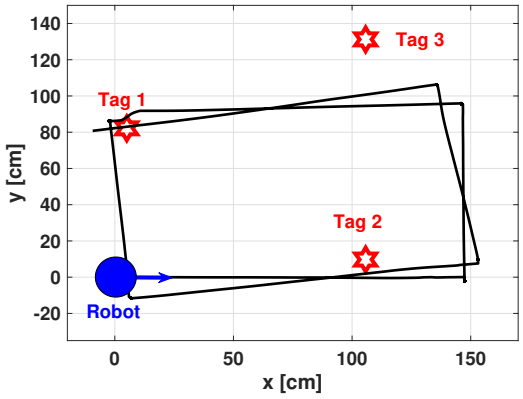


Fig. 5.8 A map of the considered environment, with the robot in its initial position and orientation (defining the global frame adopted) and the (x,y) position of the three tags considered in the experiment. The path estimated by applying the proposed SLAM algorithm is also reported.

5.1.2 Experimental results

A set of experimental tests has been performed in the office room depicted in Figure 5.7. The robot used in the experiments is a custom unicycle-like vehicle with a differential drive kinematics where the distance between the left and right wheel is 38.2 cm. The robot mounts a Raspberry Pi 4 with a Linux Ubuntu 20.04 OS where a motion planner has been developed for the high level control. An Arduino Mini is also installed in order to control the low level references to the motors and the encoder readings. The reader antenna is placed on board the robot at a height of 32 cm from the floor, it is a right-hand circularly polarized microstrip patch antenna with 7 dBi of gain. The reader (M6e ThingMagic), wireless controlled by means of a remote PC and a Raspberry Pi 4 on board the robot, supplies the antenna with a power of 25 dBm and collects measured data with a rate of 15 Hz while the robot moves with a speed of $0.2\,\mathrm{m\,s^{-1}}$. Measurements have been performed with UHF-RFID wave at the frequency of 868 MHz. Tags have right-hand circular polarized antennas similar to that of the reader, they have been realized with a stacked annular ring microstrip antenna as described in [167]. Their low profile (the thickness is less than 1cm) is well suited to be mounted on the ceiling by means of a small metallic ground plane as shown in Figure 5.7. The distance between the plane of placement of the tags and the plane of placement of the reader antenna is about 2.5 m.

The experiments have been conducted with the robot moving autonomously along a path while collecting data from the encoders and from the UHF-RFID reader. The data collected during the runs have been used offline to feed the developed algorithms and to assess their performances. A map of the environment is reported in Figure 5.8 together with the estimated robot trajectory for the experiment described in this section. Nominally the robot is requested to cover a rectangular path performing two turns in the area under the tags. The requested trajectory is followed open loop by the robot. Due to disturbances on the encoder readings and wheels slipping, the programmed trajectory (a rectangular path) differs from the real trajectory, as witnessed by the trajectory reconstructed by the SLAM algorithm. The ground truth of the overall trajectory is not available in this experiment: we only measured the final position of the robot at the end of its mission, which allowed to compute the performance indexes considered in this work. In particular, we report in Tables 5.3 and 5.4 the true and the estimated distances d_{ij} among tags ($i, j \in \{1,2,3\}$) and, respectively, the true and the estimated distance d_i of the final robot position from the projection on the floor of the three tags ($i = 1,2,3$). Figure 5.9 reports the map reconstructed by the algorithm.

The proposed methodology appears effective also in this experimental case even if the estimation errors increase with respect to the simulation scenario. This mainly depends on various unmodeled disturbances acting on the system, like, e.g., the effect of the relative

Table 5.3 True and estimated inter-tag distances (in cm).

d_{ij}	true	estimated
d_{12}	124	114
d_{13}	112	99
d_{23}	121	105

Table 5.4 True and estimated distances (in cm) of the robot from the estimated tags.

d_i	true	estimated
d_1	18	33
d_2	136	141
d_3	130	131

tag-reader position on the phase measurements, which appears to produce the most relevant perturbation. According to this effect, as also observed in the literature [168], the offset in the phase measurements cannot be considered a constant quantity, as assumed in the model. Nevertheless, since this quantity is subject to estimation, the proposed algorithm is in part able to adapt to this change and to produce an acceptable behavior. Other unmodeled phenomena include multipath effects, which produce quite wrong or even missing measurements, and systematic errors like the approximate knowledge of system parameters, including the wheel and the robot dimension. Due to these unmodeled phenomena and to other systematic sources of error, the approach in [165] was not able to produce effective estimation results in this experimental context, with the filter diverging after some steps.

5.2 Application to a visual-UWB system

In this section the results of the application of the resilient multi sensor fusion paradigm presented in Section 4.2 will be reported. The results will be presented from several sets of real-world experiments that have been conducted and whose details will be presented in Section 5.2.1.

5.2.1 Experimental results

Experimental setup

The experiments have been carried out in an indoor environment (6×8.3 m^2) cluttered with furniture, boxes (both wooden and metallic) and desks, as shown in Figure 5.10-a. The agent is equipped with an Intel RealSense D435i camera, an UWB EVB1000 board and a Laptop

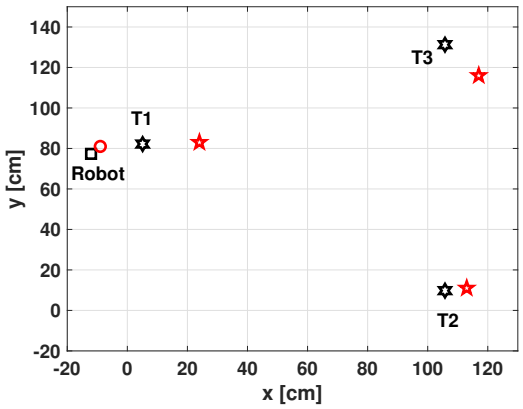


Fig. 5.9 The map realized by the proposed algorithm: black and red stars represent respectively the true and the estimated position of the three tags. The black square and the red circle represent the true and, respectively, the estimated final position of the robot. Since the ground truth for all the path is not available for this experiment, only the final robot position is depicted.

(a) The agent with two UWB antennas. **(b)** A close-up of the agent.

Fig. 5.10 System layout in the indoor environment.

with Intel Core i7-2640M CPU @ 2.80GHz, 16GB RAM as depicted in Figure 5.10-b. The agent equipment is mounted on a four-wheeled platform trolley which can be moved freely on the floor.

At the boundary of the walkable perimeter, three UWB EVB1000 boards that are the same height as the one on the agent have been placed in unidentified positions. On the aforementioned hardware, the ORB-SLAM2 method has been built in order to compute the camera position at a rate of 15 Hz while the UWB range measurements are delivered at a rate of 24 Hz. The camera trajectory before closing a VSLAM loop was employed in the tests, and the camera trajectory following loop closure was also recorded as a benchmark. With a standard variation of 0.09 mm, the UWB board placements and ground truth points have been measured with a laser distance meter. In order to assess the proposed solution in terms of comparison between the estimated, open-loop and closed-loop Visual Odometry trajectories and switching observer resilience capability in case of camera sensor fault(s), the experiments involved moving the agent around the room while recording the UWB and the camera data.

Parameter tuning

The ORB-SLAM2 algorithm has been modified in order to increase its performances (with the introduction of both computational and GPU optimizations) and to add more parameters control (open source code is available in [169]). The ORB-SLAM2 has been used in IR-D mode, using the Infrared and Depth streams from the RealSense camera at 30 Hz with a resolution of 640×480 pixels and the number of features per image has been set to 800. For the outlier detection, α-quantile χ_α is chosen from the Chi-square distribution table for 6 degrees of freedom (range and bearing of 3 antennas are considered) and with the significance level being 1 %, it is 16.81. The robust Kalman filter gain relies upon the constants a_0 and a_1, appearing in (4.31). Based on the work in [166], a_0 ranges in $[1.0, 2.5]$ and a_1 ranges in $[3.0, 8.0]$. For the experiments $a_0 = 1.5$ and $a_1 = 3.5$ have been chosen in order to achieve a good trade-off between marking an observation as an outlier and integrating it as a correct measurement. The standard deviation σ_d has been chosen as 0.05 m, as it directly depends on the maximum expected agent displacement over a sampling step.

Experiments

In the first experiment, the agent has been pulled around the room for 24.23 m while the camera and the UWB antenna mounted on the agent were acquiring data from the environment. The agent movement has been stopped in 4 points that have been marked and used as

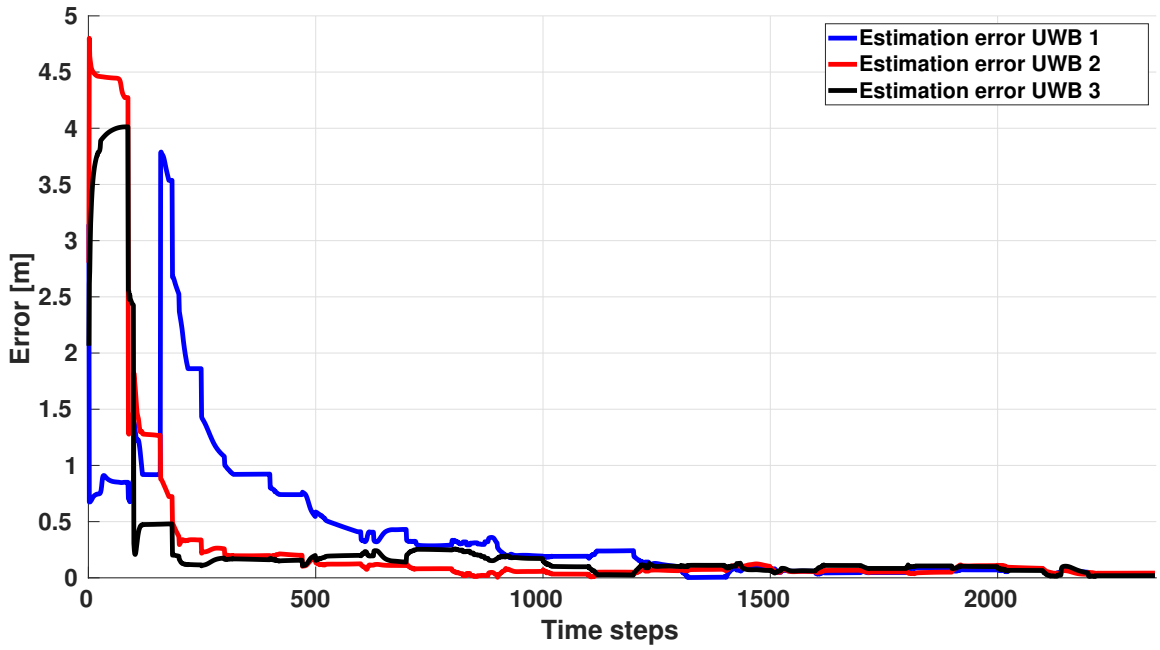


Fig. 5.11 UWB antenna position estimation errors for the first experiment.

waypoints for ground truth later on. During the experiment the open-loop Visual Odometry (OLVO) and the UWB ranges have been recorded and used respectively as the inputs $\hat{\mathbf{x}}^{vo}$ and ρ for the switching observer. The closed-loop Visual Odometry (CLVO), available as soon as the ORB-SLAM2 algorithm has closed a loop, has been recorded as well. Table 5.5 shows

Trajectory	Mean	Median	STD
CLVO	0.0279	0.0338	**0.0201**
OLVO	0.0481	0.0389	0.0288
REKF-SLAM	**0.0252**	**0.0142**	0.0325

Table 5.5 Comparison of Euclidean distances between the four ground truth waypoints and CLVO, OLVO of ORB-SLAM2 and REKF-SLAM trajectory estimates evaluated at the waypoints (in m) for the first experiment (best values are in bold).

the comparison between the CLVO, OLVO of ORB-SLAM2 and the REKF-SLAM estimate evaluated in the four waypoints used in the experiments. The table provides mean, median and standard deviation for each trajectory, showing the effectiveness of the proposed solution, which avoids trajectory drifts that may arise in VSLAM algorithms if there is no loop-closure for a long distance. Figure 5.11 shows the estimation errors for the UWB antenna locations that, after an initial phase, become very small. In the first steps, the errors are very high due to the unknown bearing and also on the particular path that the agent has traveled.

In the second experiment the agent has been pulled around the room for 22.41 m while the camera and the UWB antenna mounted on the agent were acquiring data from the environment. The agent movement has been stopped in four points that has been marked and used as waypoints as mentioned in the previous experiment. During the experiment two camera faults have been simulated preventing the system to use the open-loop Visual Odometry and thus forcing the switching observer to adapt to the new perception configuration (without any visual odometry available). The first fault occurs at about 26 s after the beginning of the movement and it lasts for about 14 s, while the second fault has been caused at about 70 s after the start and it lasts for about 8 s. Table 5.6 shows the effectiveness of the

Trajectory	Mean	Median	STD
CLVO	0.0282	0.0341	0.0205
OLVO	0.0498	0.0411	0.0325
REKF-SLAM	**0.0278**	**0.0333**	**0.0199**

Table 5.6 Comparison of Euclidean distances between the 4 ground truth waypoints and the CLVO, OLVO of ORB-SLAM2 and REKF-SLAM trajectory estimate evaluated in the waypoints (in m) for the second experiment (best values are in bold).

proposed solution, which both avoids trajectory drifts and guarantees the system resilience against camera sensor faults (i.e. the system is still able to work if the camera sensor has a malfunction). Figure 5.12 depicts the closed-loop, open-loop Visual Odometry and the estimated trajectories both during the normal operation and when the system detects camera faults. Furthermore, the trajectory with the second camera sensor fault is affected by strong multipath effect, as visible in the red dashed upper left trajectory in Figure 5.12. Nonetheless, the system is still capable of filtering the disturbances and can give an acceptable estimate of the agent trajectory[1]. As for computation times, the proposed algorithm adds a 5 % overhead to the ORB-SLAM2 algorithm when it is functioning without any fault, while its computation time is drastically reduced when a camera fault arises, performing 130 times faster than the ORB-SLAM2 algorithm (computation times comparison have been performed on the hardware platform specified in Section 5.2.1).

EuRoC Dataset Experiments

Finally, our algorithm has been validated using public datasets and real hardware experiments. In particular, the presented algorithm has been compared to VINS-Mono, VIR SLAM and ORB-SLAM2 computing the absolute trajectory error (ATE) on the EuRoC dataset [170]

[1]A video of the second experiment is available at https://youtu.be/kZd_W-vbFS0.

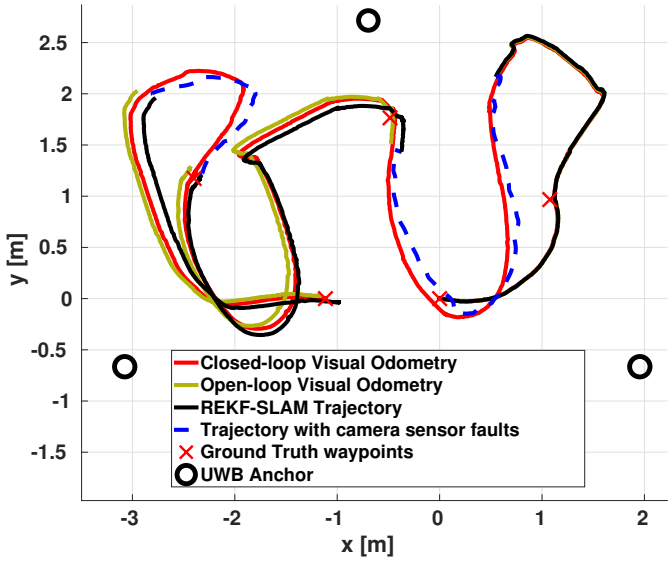


Fig. 5.12 Closed-loop, open-loop Visual Odometry, REKF-SLAM trajectory and trajectory with camera sensor faults and UWB antennas position. The ground truth waypoints are depicted with a red x.

ATE Error (m)	VINS-Mono	ORB-SLAM2 (stereo)	VIR SLAM	REKF-SLAM
EuRoC MH_01	0.186	0.035	0.178	**0.028**
EuRoC MH_02	0.240	0.018	0.188	**0.016**
EuRoC MH_03	0.271	0.028	0.260	**0.020**
EuRoC MH_04	0.402	**0.119**	0.366	**0.119**
EuRoC MH_05	0.388	0.060	0.291	**0.049**

Table 5.7 ATE comparison between VINS-Mono, ORB-SLAM2, VIR SLAM and our REKF-SLAM. The best values for ATE are presented in bold.

where ground truth is also available. The EuRoC dataset does not embed UWB ranging measurements, so a methodology to generate and integrate those data into the dataset is required. In [171] the authors suggest adding a static anchor assumed in the origin of the frame created during the robot initialization; then they added a Gaussian white noise $\mathcal{N}(0, 0.05)$ to model the error of the UWB sensor. However, modeling the error in such a way does not take into account the components related to the bias and the multipath errors of the UWB sensor. So, the strategy adopted in our experiments, in order to aggregate UWB data, is to train a deep neural network (DNN [172]) having as inputs real UWB sensor measurements, real agent and UWB positions in order to also embed information about the real error profile (comprising the Gaussian white noise, bias and multipath errors). Then we pass the ground truth data from the EuRoC datasets with the virtual UWB positions and we use the trained

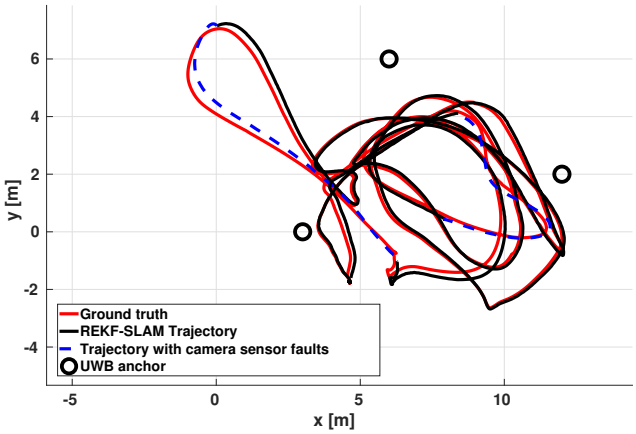


Fig. 5.13 Ground truth, REKF-SLAM trajectory, trajectory with camera sensor faults and UWB anchors position for the EuRoC MH_03 experiment where two faults have been simulated on the VO.

DNN network to generate synthetic data that have been used to feed our algorithm and to test the system with a realistic set of data (images from the EuRoC original dataset and UWB deep generated data). We would like to stress that the real UWB measurements were acquired in an environment basically different from that of EuRoC dataset; however, the measurements were acquired in an environment with dimensions similar to that of the EuRoC MH_01 to MH_05 and it has been populated with objects similar to those in the EuRoC environment. Table 5.7 shows how our REKF-SLAM algorithm outperform VINS-Mono, ORB-SLAM2 and VIR SLAM, where a single virtual UWB anchor, assumed to be in the origin of the frame created during the robot initialization and whose measurements are always available, is used. The metrics used for the comparison is ATE (Absolute Trajectory Error), defined as the root mean square error from error matrices (see [173] for further details). Furthermore, we want to show the system capability to cope with sensors failures (both when VO and UWB measurements are not available) using run MH_03 from the EuRoC dataset. In the first experiment we placed three virtual UWB anchors in $(3.0, 0.0), (6.0, 6.0), (12.0, 2.0)$ m and run our algorithm simulating two VO faults in different moments and with different duration (at minute 01 : 06 for 8 seconds and at minute 01 : 26 for 10 seconds). The results are showed in Figure 5.13, where the ground truth is reported in red, the REKF-SLAM trajectory is black when the system is working without faults and it is blue dotted when the VO is not available. Furthermore, the ATE has been computed for the presented experiment being 0.045 m; the ATE is slightly higher than that of ORB-SLAM2 in EuRoC MH_03 and this is something reasonable given that the system for 18 s in the experiments is running without the VO and uses the UWB measurements only that are affected by noise, as described in the previous paragraph. Still, the ATE is acceptable and comparable to the results presented in Table 5.7.

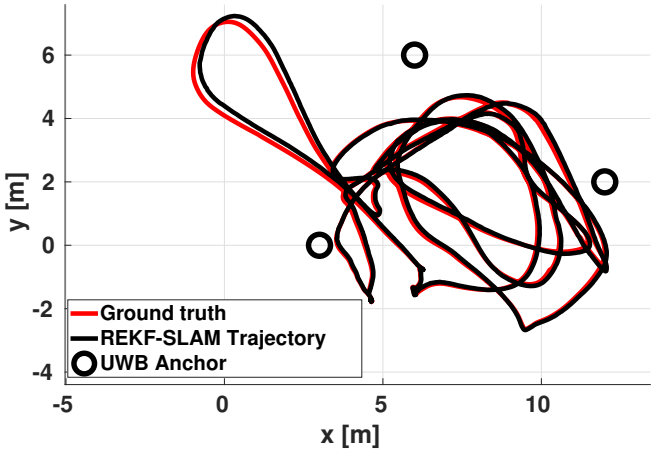


Fig. 5.14 Ground truth, REKF-SLAM trajectory and UWB anchors position for the EuRoC MH_03 experiment where the UWB maximum range has been set to 7 m.

The last experiment has been conducted using the EuRoC MH_03 dataset, with three virtual UWB anchors in $(3.0, 0.0), (6.0, 6.0), (12.0, 2.0)$ m. Here the UWB maximum reading range has been set to 7 m in order to simulate conditions where the UWB measurements are not available for a variable amount of time and where sometimes all the three UWB anchors are not available at all. Figure 5.14 shows the results of the experiment; the most challenging situation is in the upper left part of the figure, where the agent is further than 7 m from the three UWB anchors. Here the REKF-SLAM trajectory and ground truth differs more than in the rest of the path since the UWB measurements are not available and the trajectory estimation slightly degrades. However, the presented algorithm is able to maintain a low ATE that, for this last experiment, is 0.025 m still lower than that of ORB-SLAM2 (0.028 m).

5.3 Application to a visual-UWB system with deep learning approach

In this section the experiments carried out on a system based on visual odometry and UWB range measurements with deep learning approach are reported. The experiments refer to the multi sensor fusion paradigm presented in Section 4.3. The results will be presented from real-world acquired measurements used to train the deep neural network on a simulation environment in Section 5.3.1. These results have been compared to those obtained in Section 5.2.1.

5.3.1 Simulations

Dataset for training

In order to provide a dataset for the training of the deep neural network model presented in Section 4.3.1, we resorted to the following setup: the indoor environment where the acquisitions of the sensor measurements have been run is a 6×8.3 m^2 space cluttered with furniture, boxes (both wooden and metallic) and desks (similar to that shown in Figure 5.10-a). The agent is equipped with an Intel RealSense D435i camera, an UWB EVB1000 board and a Laptop with Intel Core i7-2640M CPU @ 2.80GHz, 16GB RAM as depicted in Figure 5.10-b. The agent equipment is mounted on a four-wheeled platform trolley which can be moved freely on the floor.

At the boundary of the walkable perimeter, three UWB EVB1000 boards that are the same height as the one on the agent have been placed in unidentified positions. On the aforementioned hardware, the ORB-SLAM2 method has been built in order to compute the camera position at a rate of 15 Hz while the UWB range measurements are delivered at a rate of 24 Hz. The trajectory estimated by the ORB-SLAM2 algorithm after a loop closure has been used as ground truth.

The dataset for training the deep neural network is composed of 90000 samples (for each input and output): 52 multiple acquisitions of about 5 min long time-series for the following inputs $x_r, y_r, z_r, x_{u_i}, y_{u_i}, z_{u_i}$ and for the output ρ (UWB range measurement).

Results

The simulations have been carried out in a simulated environment where three UWB antennas have been placed in the area (6×6 m^2) and the results have been compared to those obtained in Section 5.2.1 in terms of trajectory estimation quality (RMSE) and in estimation of the UWB antenna locations. A set of 1000 simulations comprising 3000 steps has been run in the simulation environment, with a fixed location for three UWB antennas in position $(3,6,0)$, $(6,0,0)$, $(0.2,2,0)$ m, while changing the agent trajectory in each simulation, whose average length is about 30 m long. Here we want to underline that the results from Section 5.2.1 have been obtained using an analytical model that does not take into account some physical phenomena (like bias, multi-path, etc.). The results of a simulation, in terms of agent trajectory, are reported in Figure 5.15. From this picture we can assess that the agent estimated trajectory with the method presented in this section is much closer to the ground truth trajectory compared to the trajectory estimated with the analytical model presented in Section 5.2.1. Furthermore, the Root Square Error between the real and estimated trajectories at each time-step is reported in Figure 5.16, showing how the method based on the

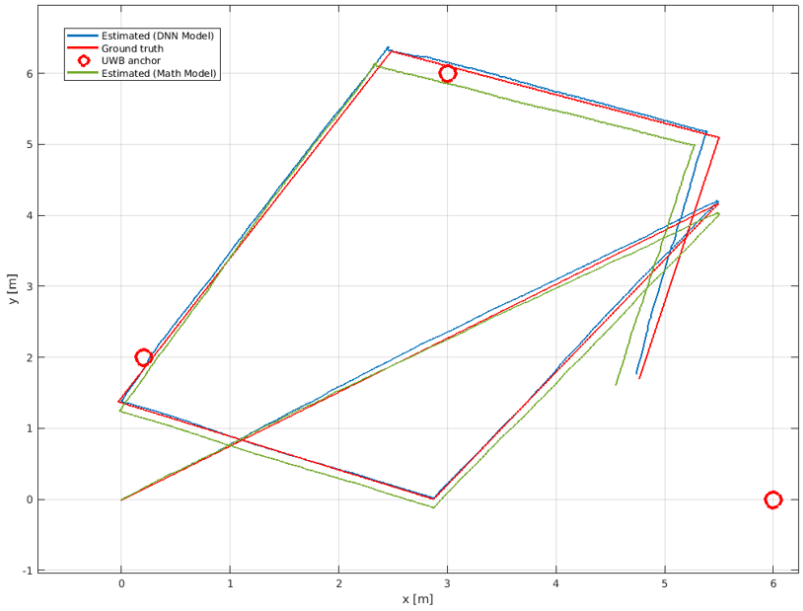


Fig. 5.15 Indoor scenario with three UWB antennas located in position $(3,6,0)$, $(6,0,0)$, $(0.2,2,0)$ depicted as red circles in the picture. The ground truth trajectory is depicted in red, the trajectory estimated with the math model (Section 5.2.1) is depicted in green, while the trajectory estimated with the DNN measurement model presented in this section is depicted in blue.

DNN measurement model presents better results, reducing the RSE along almost the entire trajectory (apart from the steps ranging in $[250,500]$, where the RSE of the other method performs better). Finally, we present the results of the estimation of the three UWB antenna positions in Figure 5.17. From this picture it can be seen that the steady-state error between the estimated and real position of the 3 UWB antennas is lower in the case of the DNN-based measurement model, compared to the analytical model.

5.4 Application to a multiple visual system

In this section the results of the application of the multiple visual sensor fusion paradigm presented in Section 4.4 will be reported. The results will be presented from several set of real-world experiments that have been conducted and whose details will be presented in Section 5.4.1.

5.4.1 Experimental results

A number of experiments have been used to quantify the performance and capabilities of the sensor fusion system. The experiments were conducted in a facility that was 16×12

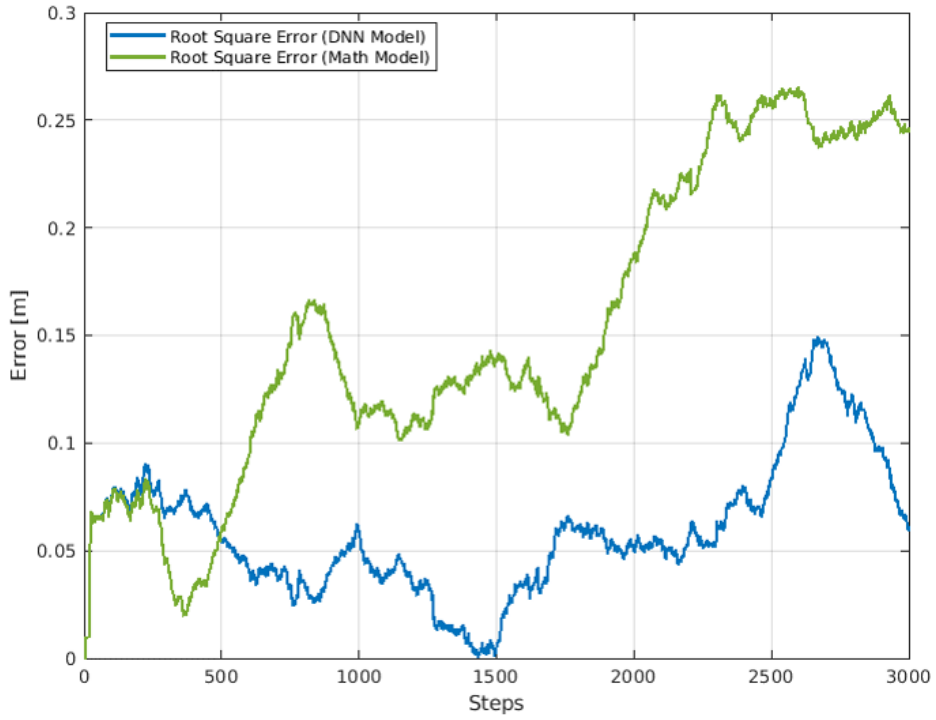


Fig. 5.16 RSE comparison between DNN and analytical measurement models. The Root Square Error at each time step for the DNN measurement model (presented in this section) is depicted in blue, for the analytical measurement model (from Section 5.2.1) is depicted in green.

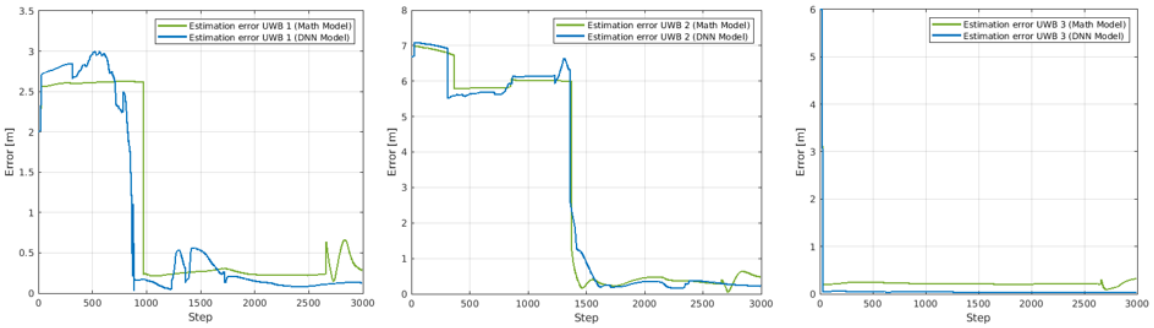


Fig. 5.17 Position estimation error of UWB antenna positions between DNN and analytical measurement models. The error at each time step for the DNN measurement model (presented in this section) is depicted in blue, for the analytical measurement model (from Section 5.2.1) is depicted in green.

m in size and filled with instructive items to aid vision-based localization algorithms. The system's accuracy has been assessed on a discrete trajectory specified by particular waypoints at which measurements were collected due to a lack of sufficient equipment, such as a Vicon system. Along the desired route, the waypoints that indicate the actual location were inserted. A Hanmatek laser rangefinder with a ± 1.5 mm precision was used to locate them. The experiments were conducted in two stages: the first phase focused at gathering data to fine-tune the spatial correction algorithm's parameters, and the second phase aimed at verifying the entire sensor fusion system. The training phase was conducted in the manner depicted in Figure 5.18. In order to obtain the estimated pose of the individual systems at the waypoints 3, 4, 5, 7, and 9 — the most instructive points along the entire path — and because the accuracy of visual-based algorithms increases after loop closures, a number of experiments were conducted. Figure 5.18 illustrates how the systems are impacted by spatial expansion, as the accumulated error increases in proportion to the distance from the starting point along both axes, using tracks estimated by ORB-SLAM2, ZED Mini, and the PX4 EKF2 with the previous estimates blended with $\alpha_1, \alpha_2 = 0.5$ as input. The following regressors parameters were determined from the training phase data: $(\beta_{x,0}, \beta_{x,1}, \beta_{y,0}, \beta_{y,1}) = (0.9953, -0.0044, 0.9901, -0.0040)$. In order to assess the system's real performance, a different path from that used for training was used during the validation phase. That validation path, as shown in Figure 5.19, has been completed with a different direction of travel, which, for visual-based algorithms, generates a different behavior. It also contains additional places that have not been previously observed, namely 12, 13, and 14, and 15. Figure 5.19 qualitatively demonstrates that while the track estimated by the entire fusion system is more precise, the systems individually behave as shown in Figure 5.18. The RMSE between the locations of the waypoints and those predicted by the systems is shown in Table 5.8.

The robustness to potential errors is another property of the suggested sensor fusion system, and a second test was conducted to confirm this. Figure 5.20 illustrates how the failure and reset of ORB-SLAM2 was caused during several legs, i.e. $3 \rightarrow 4$, $14 \rightarrow 12$, and $9 \rightarrow 7$, to demonstrate how the sensor fusion system continues to function and accurately predicts the location with a single sensor.

Waypoint	Ground truth (x, y) [m]	ORB-SLAM2 (x, y) [m]	ZED Mini (x, y) [m]	Fused (x, y) [m]
3	(2.053, 5.634)	(1.952, 5.752)	(2.042, 5.786)	(2.056, 5.677)
4	(2.053, 1.676)	(2.059, 1.718)	(1.996, 1.837)	(2.085, 1.778)
5	(4.788, 7.901)	(4.737, 8.155)	(4.821, 8.046)	(4.780, 7.889)
7	(7.826, 8.501)	(7.911, 8.844)	(7.968, 8.683)	(7.868, 8.510)
9	(12.986, 10.626)	(13.396, 11.202)	(13.269, 10.816)	(12.956, 10.581)
14	(10.242, 0.900)	(10.524, 0.905)	(10.382, 0.732)	(10.269, 0.825)
RMSE		0.361	0.196	**0.034**

Table 5.8 RMSE between the estimates (ORB-SLAM2, ZED Mini and fused) and the ground truth waypoints; the best results are reported in bold.

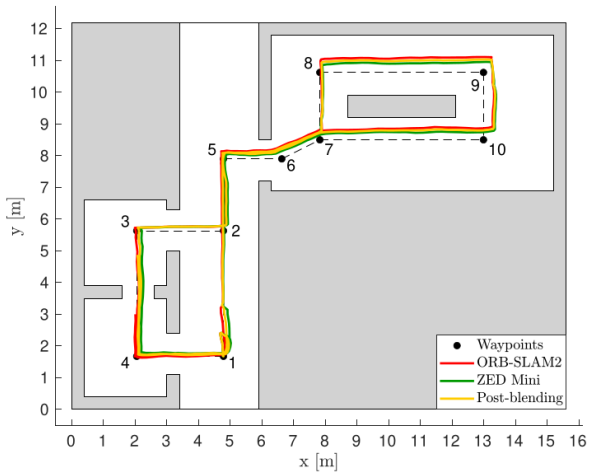


Fig. 5.18 Trajectories estimated using ORB-SLAM2 (red), ZED Mini (green), and sensor fusion system up to the blending stage (yellow) over the training path (black, dashed) defined by the following waypoints: 1-2-5-6-7-8-9-10-7-6-5-2-3-4-1.

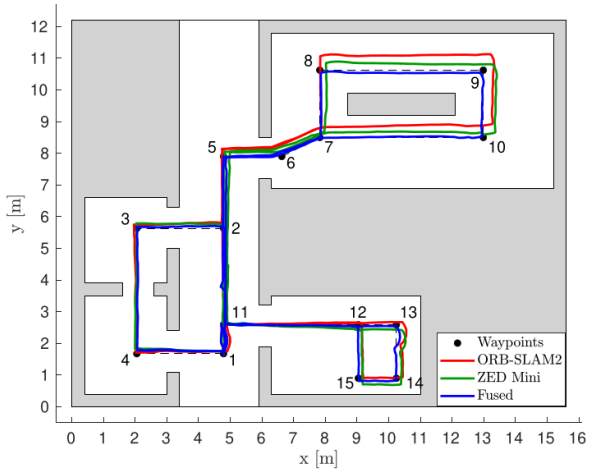


Fig. 5.19 Trajectories estimated using ORB-SLAM2 (red), ZED Mini (green), and full sensor fusion system (blue) over the validation path (black, dashed) defined by the following waypoints: 1-4-3-2-11-12-15-14-13-12-11-5-6-7-10-9-8-7-6-5-2-11-1.

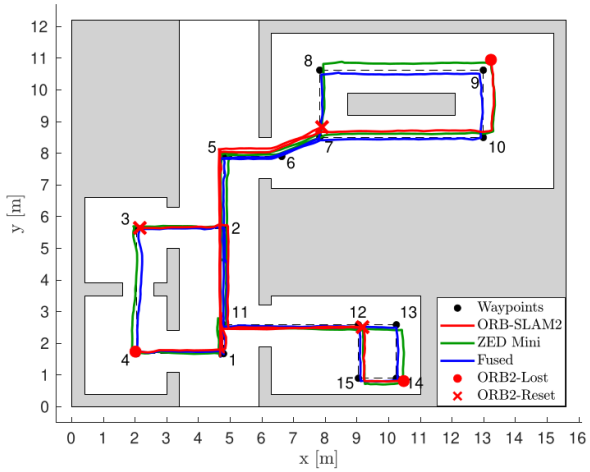


Fig. 5.20 Trajectories estimated using ORB-SLAM2 (red), ZED Mini (green), and full sensor fusion system (blue) over the validation path (black, dashed); the ORB-SLAM2 system has been forced to reset multiple times to test sensor fusion resilience to the loss of a sensor.

Chapter 6

Impact of research: resulting publications

This thesis presented algorithms, methodologies, and use cases to provide some answers to the hypothesis using a multi-sensor fusion approach that uses both classical and deep learning-based techniques to achieve resilient perception. Peer reviewed publications authored as a result of research in this thesis are presented below to show the research impact of this thesis, for a total of **18** papers. Note that Journal Papers are presented in **bold typeface** and conferences in normal typeface.

1. **Sensor resilience for localization**: Proposed an approach to localization for autonomous systems (unicycle-like robots) based on classical methodologies (e.g. EKF). The sensor inputs range from UHF-RFID to UWB range sensors fusing encoders mounted on the wheels. The designed system integrates sensor resilience.
   Resulting publications:

   a) Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. A localization system for autonomous vehicles based on trilateration tags. In 2022 16th European Conference on Antennas and Propagation (EuCAP), pages 1–5, 2022. [2]

   b) Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Considering polarization mismatch in modeling the rfid phase offset variability for tag localization. In 2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA), pages 21–24, 2022. [15]

2. **Sensor resilience for SLAM**: Proposed an approach to solve the Simultaneous Localization And Mapping for autonomous systems (unicycle-like robots and autonomous agents) based on classical methodologies (e.g. EKF). The sensor inputs range from UHF-RFID to UWB range sensors fusing encoders and visual odometry systems

mounted on the system. The designed system integrates sensor resilience.
Resulting publications:

a) Francesco Martinelli and Fabrizio Romanelli. A slam algorithm based on range and bearing estimation of passive uhf-rfid tags. In 2021 IEEE International Conference on RFID Technology and Applications (RFID-TA), pages 20–23, 2021. [1]

b) Francesco Martinelli and Fabrizio Romanelli. A resilient ro-slam algorithm with bearing reconstruction of detected landmarks. In 2022 3rd International Conference on Artificial Intelligence, Robotics and Control (AIRC), pages 61–66, 2022. [3]

c) **Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Robust simultaneous localization and mapping using the relative pose estimation of trilateration uhf rfid tags. IEEE Journal of Radio Frequency Identification, 6:583–592, 2022.** [4]

d) **Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. The role of the rfid polarization mismatch in the simultaneous localization and mapping problem. IEEE Journal of Radio Frequency Identification, 2023.** [5]

e) Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Exploiting the orientation of trilateration uhf rfid tags in robot localization and mapping. In 2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA), pages 5–8, 2022. [6]

f) **Fabrizio Romanelli, Francesco Martinelli, and Emidio Di Giampaolo. Robust simultaneous localization and mapping using range and bearing estimation of radio ultra high frequency identification tags. IEEE Transactions on Control Systems Technology, 31(2):772–785, 2023.** [7]

g) **Francesco Martinelli, Simone Mattogno, and Fabrizio Romanelli. A resilient solution to range-only slam based on a decoupled landmark range and bearing reconstruction. Robotics and Autonomous Systems, 160:104324, 2023.** [8]

h) Fabrizio Romanelli, Francesco Martinelli, and Emidio Di Giampaolo. Robust simultaneous localization and mapping using range and bearing estimation of radio ultra high frequency identification tags. 2023 7th IEEE Conference on Control Technology and Applications (CCTA 2023), 2023. [9]

i) **Fabrizio Romanelli, Francesco Martinelli, and Simone Mattogno. Resilient simultaneous localization and mapping fusing ultra wide band range measurements and visual odometry. Journal of Intelligent & Robotic Systems, 109(64), 2023.** [10]

3. **Improvement of sensing abilities**: Developed a methodology to increase the sensing abilities fusing several different visual odometry methodologies. The developed algorithms have been tested and applied on UAVs. In the same context, a distributed architecture for UASs based on ROS 2 has been designed and proposed.
   Resulting publications:

   a) Lorenzo Bianchi, Daniele Carnevale, Roberto Masocco, Simone Mattogno, Federico Oliva, Fabrizio Romanelli, and Alessandro Tenaglia. Efficient visual sensor fusion for autonomous agents. In 7th International Conference on Control, Automation and Diagnosis (ICCAD'23). IEEE, 2023. [11]

   b) **Lorenzo Bianchi, Daniele Carnevale, Fabio Del Frate, Roberto Masocco, Simone Mattogno, Fabrizio Romanelli, and Alessandro Tenaglia. A novel distributed architecture for unmanned aircraft systems based on robot operating system 2. IET Cyber-Systems and Robotics, 5(1):e12083, 2023.** [12]

   c) Alessandro Navone, Fabrizio Romanelli, Marco Ambrosio, Mauro Martini, Simone Angarano, and Marcello Chiaberge. Lavender autonomous navigation with semantic segmentation at the edge. In Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2023. [13]

4. **Sensor data processing and generation**: Developed several methodologies to process sensor data exploiting physical characteristics and phenomena (e.g. polarization mismatch for RFID). Design and development of methodologies based on deep neural networks (DNN) to generate realistic sensor data to be exploited in the experimental setups.
   Resulting publications:

   a) **Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Exploiting polarization mismatch to estimate the orientation of rotating uhf rfid tags. IEEE Journal of Radio Frequency Identification, 2023.** [14]

   b) Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Exploiting the electromagnetic coupling to estimate the close motion of uhf-rfid tagged objects. In 2023 IEEE 13th International Conference on RFID Technology and Applications (RFID-TA), 2023. [16]

c) **Fabrizio Romanelli and Francesco Martinelli. Synthetic Sensor Data Generation Exploiting Deep Learning Techniques and Multimodal Information. IEEE Sensors Letters, vol. 7, no. 7, pp. 1-4, July 2023, Art no. 7003404, doi: 10.1109/LSENS.2023.3290209.** [17]

d) **Fabrizio Romanelli and Francesco Martinelli. Synthetic sensor measurement generation with noise learning and multi-modal information. IEEE Access, 2023, 11:111765–111788, 2023.** [18]

Chapter 7

Societal implications

The research presented in this Ph.D. dissertation has far-reaching implications for society. Autonomous systems, such as self-driving cars, robots, and surveillance systems, are becoming increasingly prevalent in our daily lives. As these systems rely heavily on perception to interact with and navigate through the world, the advancements in perception technology brought about by this research have significant societal impacts in several key areas. These areas will be presented in this chapter. The importance of societal implications in technical research has been highlighted in [174], in [175] and in [176]. Another interesting survey focusing on the technology foresight for social good is reported in [177].

7.1 Enhancing safety and reliability

One of the most critical societal benefits of this research is the potential to enhance safety and reliability in autonomous systems. As these systems become more integrated into transportation, healthcare, and other sectors, the ability to make accurate real-time decisions based on a robust understanding of the environment is paramount. The fusion of sensor data, both classical and deep learning-based, improves perception accuracy, reducing the risk of accidents and ensuring the safety of individuals who interact with autonomous systems.

7.1.1 Reduction in accidents

The primary societal benefit of improving perception technology in autonomous systems is the significant reduction in accidents and related injuries. Autonomous vehicles, for instance, have the potential to greatly reduce the number of accidents caused by human errors, which are responsible for the majority of traffic accidents today. By fusing data from multiple sensors and leveraging both classical and deep learning techniques, these vehicles can better

understand their surroundings, anticipate potential hazards, and make split-second decisions to avoid collisions. This not only saves lives but also reduces the economic costs associated with accidents, including medical expenses, property damage, and lost productivity.

7.1.2 Improved pedestrian and cyclist safety

Pedestrians and cyclists are among the most vulnerable road users, and their safety is a paramount concern. Advanced perception systems can significantly enhance safety for these groups by accurately detecting and predicting their movements. For example, autonomous vehicles equipped with robust perception can recognize pedestrians and cyclists even in low-light conditions or when they are partially obscured by obstacles. This heightened awareness can prevent accidents and create a safer environment for vulnerable road users.

7.1.3 Emergency response and disaster management

Beyond transportation, the improvement in perception technology has implications for emergency response and disaster management. Autonomous drones and robots equipped with enhanced perception capabilities can be deployed in disaster-stricken areas to assess damage, locate survivors, and even deliver medical supplies. Their ability to navigate challenging and hazardous environments with accuracy and reliability can expedite rescue operations and ultimately save lives.

7.1.4 Healthcare applications

The impact of improved perception technology extends to the healthcare sector. Surgical robots, for instance, benefit from precise perception systems that enable them to perform minimally invasive surgeries with greater precision. These systems can reduce the risk of complications and improve patient outcomes. Additionally, autonomous healthcare delivery robots can navigate hospital environments safely, reducing the risk of cross-contamination and enhancing the efficiency of healthcare services.

7.1.5 Aging population and accessibility

As the global population ages, there is a growing need for solutions that enable older individuals to maintain their independence. Autonomous systems with advanced perception capabilities can play a crucial role in assisting the elderly with tasks such as transportation, medication management, and daily living activities. By ensuring the safety and reliability of

these systems, society can empower older individuals to age in place comfortably and with dignity.

7.2 Enabling autonomous mobility

Autonomous vehicles have the potential to revolutionize transportation, making it more efficient and accessible. By improving perception accuracy and resilience to sensor failures, this research contributes to the development of self-driving cars that can operate safely in diverse weather conditions, busy urban environments, and even in emergency situations. This can lead to reduced traffic congestion, fewer accidents, and increased mobility for individuals who cannot drive due to disabilities or other reasons.

7.2.1 Revolutionizing transportation

One of the most noticeable and transformative societal impacts of improved perception technology is its role in revolutionizing transportation through autonomous vehicles. These vehicles have the potential to reshape urban mobility by providing safer, more efficient, and accessible transportation options.

- Reduced Congestion: Autonomous vehicles can communicate with each other and traffic infrastructure to optimize traffic flow, reducing congestion in urban areas. This, in turn, decreases travel times, fuel consumption, and greenhouse gas emissions, leading to cleaner and more sustainable cities.
- Accessibility: Autonomous mobility can significantly improve accessibility for individuals who are unable to drive due to age, disability, or other reasons. Elderly individuals and people with disabilities can gain greater independence and access to essential services, such as healthcare, without relying on others for transportation.
- Ride-Sharing and Mobility as a Service (MaaS): Improved perception technology is pivotal for the success of ride-sharing and MaaS platforms. Autonomous vehicles can be seamlessly integrated into these services, offering convenient and cost-effective transportation options. This can lead to reduced car ownership, lower traffic congestion, and a shift towards more sustainable transportation models.

7.2.2 Urban planning and infrastructure

Enhanced perception technology also has a profound impact on public transportation systems. Autonomous buses and shuttles can offer efficient and flexible transit options for both urban

and rural areas. These systems can adapt to passenger demand in real time, reducing the need for fixed routes and schedules. Consequently, public transportation becomes more accessible and cost-effective, which benefits both urban and underserved communities.

7.2.3 Freight and logistics

Beyond passenger transportation, improved perception technology enables significant advancements in the logistics and freight industry. Autonomous trucks equipped with robust perception systems can operate 24/7, increasing the efficiency of goods transportation. This leads to reduced shipping costs, faster deliveries, and potentially lower prices for consumer goods. Additionally, the ability to navigate complex logistics centers autonomously can streamline supply chains, improving the availability of essential goods.

7.2.4 Environmental impact

The societal benefits of enhanced autonomous mobility extend to the environment. By optimizing routes and reducing traffic congestion, autonomous vehicles can significantly reduce greenhouse gas emissions and air pollution. This contributes to cleaner air, improved public health, and a reduction in the societal costs associated with environmental degradation and climate change.

7.2.5 Economic opportunities

The development and deployment of autonomous mobility technologies create economic opportunities in research, development, manufacturing, and service industries. This can lead to the creation of high-skilled jobs and stimulate economic growth. Furthermore, by improving transportation efficiency, businesses can reduce operating costs and potentially pass on savings to consumers.

7.3 Advancing robotics

In the realm of robotics, the research presented here has the potential to enable robots to operate more effectively in dynamic and unstructured environments. Robust perception is crucial for robots in fields such as manufacturing, agriculture, and healthcare, where they need to adapt to changing conditions and collaborate with humans. Improved perception can lead to more efficient and safer industrial processes and improved healthcare outcomes.

7.3.1 Industrial automation and efficiency

One of the significant societal impacts of enhancing perception technology in robotics is the advancement of industrial automation. Manufacturing and production industries can greatly benefit from robots equipped with robust perception systems. These robots can precisely and safely perform tasks such as assembly, quality control, and material handling. As a result, manufacturing processes become more efficient, leading to higher productivity, reduced production costs, and increased competitiveness in the global market.

7.3.2 Agriculture and food production

The agriculture sector is also poised to benefit from advanced perception technology in robotics. Autonomous agricultural robots equipped with sophisticated perception capabilities can perform tasks like planting, harvesting, and monitoring crop health with unprecedented precision. This leads to increased crop yields, reduced reliance on chemical inputs, and more sustainable farming practices. Moreover, it addresses the challenge of labor shortages in agriculture and ensures a stable food supply for society.

7.3.3 Healthcare and assisted living

Robotics in healthcare is a promising field with numerous societal benefits. Robots with advanced perception systems can assist medical professionals in tasks such as surgery, patient care, and drug dispensing. These robots enhance the precision of medical procedures, reduce the risk of human errors, and improve patient outcomes. In assisted living environments, robots can assist the elderly and individuals with disabilities with daily tasks, promoting independence and improving their quality of life.

7.3.4 Search and rescue

In emergency situations, such as natural disasters or building collapses, robots with advanced perception capabilities can be deployed for search and rescue operations. These robots can navigate through hazardous environments, locate survivors, and provide real-time information to rescue teams. This significantly improves the effectiveness of emergency response efforts and increases the chances of saving lives.

7.3.5 Environmental monitoring and conservation

Advanced perception technology in robotics also has applications in environmental monitoring and conservation. Autonomous drones and underwater robots equipped with precise sensors can collect data on ecosystems, climate change, and wildlife behavior. This information is vital for making informed decisions related to environmental protection and resource management.

7.3.6 Education and research

Robotics equipped with advanced perception systems are valuable tools in education and research. They allow researchers and students to explore complex concepts in fields such as artificial intelligence, computer vision, and machine learning. Additionally, these robots can serve as educational tools for teaching STEM (science, technology, engineering, and mathematics) subjects to students of all ages, thereby fostering a future-ready workforce.

7.3.7 Labor market impact

The integration of advanced perception technology in robotics raises questions about the labor market. While it can lead to the automation of some tasks, it can also create new job opportunities in fields related to robotics development, maintenance, and supervision. Society must address the need for reskilling and upskilling the workforce to adapt to the changing employment landscape.

7.4 Enhancing security and surveillance

Autonomous surveillance systems play a crucial role in public safety and security. By improving perception accuracy and resilience, this research can contribute to the development of surveillance systems that can better detect and respond to security threats. This is particularly relevant in areas such as border control, airport security, and public event monitoring, where rapid and accurate threat detection is essential.

7.4.1 Improved threat detection

Enhanced perception technology in security and surveillance systems is paramount for improving threat detection capabilities. It allows for more accurate and timely identification of potential security threats, including intruders, suspicious objects, and unauthorized access.

This heightened level of threat detection is vital for protecting critical infrastructure, public spaces, and private property.

7.4.2 Public safety

In public spaces such as airports, train stations, and shopping malls, advanced perception technology enhances public safety. Surveillance systems with improved perception capabilities can monitor large crowds and identify potential security risks or emergencies. This proactive approach to public safety can lead to rapid response times and the prevention of incidents that could endanger lives.

7.4.3 Counterterrorism and law enforcement

The fight against terrorism and crime benefits significantly from advanced perception technology. Surveillance systems equipped with precise sensors and machine learning algorithms can identify suspicious behaviors and individuals, aiding law enforcement in their efforts to prevent and investigate criminal activities. These technologies provide a force multiplier effect, allowing law enforcement agencies to cover larger areas and respond more effectively.

7.4.4 Critical infrastructure protection

Critical infrastructure, such as power plants, water treatment facilities, and transportation hubs, is vulnerable to various threats. Enhanced security and surveillance systems with advanced perception capabilities help protect these assets from physical and cyber threats. The ability to detect and respond to security breaches promptly is crucial for ensuring the continuity of essential services and safeguarding national interests.

7.4.5 Privacy considerations

While advanced perception technology enhances security and surveillance, it also raises important privacy considerations. Striking the right balance between security and privacy is a societal challenge. Ethical deployment of surveillance technology, clear policies, and transparency are essential to maintain public trust and protect individual privacy rights.

7.4.6 Crime deterrence

The presence of surveillance systems with advanced perception capabilities can act as a deterrent to criminal activity. Potential wrongdoers are less likely to engage in illegal

activities when they know they are being monitored. This deterrence effect contributes to the overall reduction in crime rates, creating safer communities.

7.4.7 Emergency response

In emergency situations, such as natural disasters or large-scale accidents, advanced perception technology can provide critical information to first responders. Surveillance systems equipped with precise sensors can assess the extent of damage, locate survivors, and guide emergency response efforts, thereby saving lives and reducing the impact of disasters on communities.

7.5 Economic impact

The widespread adoption of autonomous systems can have a significant economic impact. Improved perception can lead to cost savings through reduced accidents, increased productivity, and more efficient resource utilization. Additionally, the development and deployment of advanced perception technologies can stimulate economic growth by creating job opportunities in research, development, and manufacturing.

7.5.1 Job creation and skills development

The advancement of perception technology in autonomous systems creates new job opportunities across various sectors. Research and development in robotics, artificial intelligence, and machine learning are essential components of this progress. Engineers, data scientists, software developers, and technicians are in high demand to design, develop, maintain, and operate these sophisticated systems. As a result, the job market expands, and individuals have the opportunity to pursue rewarding careers in cutting-edge fields.

7.5.2 Manufacturing and supply chain efficiency

The integration of advanced perception technology in manufacturing and logistics leads to increased efficiency and reduced production costs. Automation and robotics systems equipped with precise perception capabilities can operate around the clock with minimal downtime. This efficiency translates into cost savings for businesses, which can be passed on to consumers through lower prices or invested in research and development to drive further innovation.

7.5.3 Market growth and innovation

Improved perception technology drives innovation in various industries. As autonomous systems become more capable and reliable, they enable the development of entirely new products and services. This innovation not only expands market opportunities but also fosters competition, leading to more choices and better-quality products for consumers.

7.5.4 Global competitiveness

Nations that invest in and adopt advanced perception technology gain a competitive edge in the global marketplace. Businesses that leverage these technologies can deliver higher-quality products and services, often at a lower cost, making them more competitive internationally. This, in turn, contributes to economic growth and increased exports.

7.5.5 Entrepreneurship and startups

The accessibility of advanced perception technology and the availability of data-driven solutions create fertile ground for entrepreneurship and startups. Innovators and entrepreneurs can identify new market niches and develop solutions to address them. This entrepreneurial ecosystem fosters creativity, job creation, and economic dynamism.

7.5.6 Economic resilience

During economic downturns or disruptions, businesses equipped with advanced perception technology may exhibit greater resilience. The automation and efficiency gains realized through these technologies can help companies weather economic challenges by reducing operational costs and maintaining productivity even in adverse conditions.

7.5.7 Cross-industry collaboration

The development of advanced perception technology often involves cross-industry collaboration. Academia, government agencies, and private sector companies collaborate on research and development projects. This collaborative environment enhances knowledge sharing, accelerates technological progress, and contributes to economic growth.

7.5.8 Infrastructure investment

Governments and private sector entities often invest in infrastructure to support advanced perception technology. This includes the deployment of sensors, communication networks,

and data centers. Such investments create construction jobs and stimulate economic activity in related sectors.

7.5.9 Increased access to education and training

The demand for skilled professionals in fields related to perception technology leads to increased access to education and training programs. Educational institutions offer courses, certifications, and degree programs in artificial intelligence, robotics, and data science, enabling individuals to acquire the skills needed for high-demand jobs.

7.6 Ethical considerations

As autonomous systems become more integrated into society, ethical considerations become increasingly important. The research presented here also addresses some of these ethical concerns. The ability to detect and respond to sensor failures, for example, is crucial for ensuring the responsible use of autonomous systems and mitigating the potential for harm.

7.6.1 Privacy and surveillance

One of the most significant ethical considerations related to advanced perception technology is the balance between security and privacy. The widespread use of surveillance systems with enhanced perception capabilities can raise concerns about the potential invasion of privacy. It is crucial to establish clear policies and regulations that govern how data collected by these systems is used, stored, and shared. Additionally, the implementation of privacy-preserving technologies, such as anonymization and encryption, can help mitigate privacy risks.

7.6.2 Bias and fairness

Advanced perception technology, particularly those powered by machine learning algorithms, can inadvertently perpetuate bias and discrimination. If training data used to develop these systems contains biases, the technology may produce unfair or discriminatory outcomes. Ethical considerations include addressing bias in algorithms, ensuring fairness in decision-making processes, and promoting transparency in how these systems work.

7.6.3 Accountability and liability

As autonomous systems become more sophisticated, questions arise regarding accountability and liability in the event of accidents or unintended consequences. Determining who is responsible when an autonomous vehicle is involved in a collision, for instance, raises complex ethical and legal questions. Developing clear frameworks for assigning responsibility and liability is essential to ensure that individuals and organizations are held accountable for their actions.

7.6.4 Safety and reliability

Ethical considerations also encompass the safety and reliability of autonomous systems. Ensuring that these systems are designed to prioritize safety and minimize risks is paramount. This includes implementing fail-safe mechanisms, robust testing procedures, and continuous monitoring to prevent accidents and minimize harm.

7.6.5 Transparency and explainability

Transparency and explainability are essential ethical principles in the context of perception technology. End-users and stakeholders should have a clear understanding of how these systems make decisions and interpret sensor data. Explainable AI (XAI) methods are being developed to make machine learning models more transparent and interpretable, allowing users to trust and validate their decisions.

7.6.6 Human autonomy and control

Ethical concerns extend to the degree of human autonomy and control in autonomous systems. There is a need to strike a balance between automation and human decision-making. Ensuring that humans can intervene and override autonomous systems in critical situations is an ethical imperative to prevent undue reliance on technology.

7.6.7 Data security and cybersecurity

The protection of data and systems from cyberattacks and data breaches is an ethical obligation. Advanced perception technology relies heavily on data collection and processing, making it vulnerable to malicious actors. Robust cybersecurity measures, including encryption, regular security audits, and secure data storage, are necessary to safeguard sensitive information and prevent unauthorized access.

7.6.8 Environmental impact

Ethical considerations extend to the environmental impact of perception technology. The production and disposal of hardware components, energy consumption, and electronic waste management are areas where ethical choices can influence sustainability. Sustainable practices, such as using eco-friendly materials and optimizing energy usage, align with ethical responsibilities to protect the environment.

7.6.9 Accessibility and inclusivity

Ethical considerations also encompass ensuring that advanced perception technology is accessible and inclusive for all individuals, including those with disabilities. Designing systems that accommodate diverse needs and providing equitable access to the benefits of these technologies are ethical imperatives.

7.6.10 Ethical governance and regulation

Establishing ethical governance and regulatory frameworks is essential to address the evolving ethical challenges posed by advanced perception technology. Governments, industry stakeholders, and ethical advisory boards play a crucial role in setting standards, enforcing regulations, and ensuring responsible development and deployment of these technologies.

7.7 Final considerations

In conclusion, the research on multi-sensor fusion for autonomous resilient perception has profound societal impacts, ranging from improved safety and mobility to advancements in various industries and economic growth. However, it also comes with ethical responsibilities, and it is essential to continue exploring how these technologies can be deployed responsibly and in ways that benefit society as a whole. This research contributes to the ongoing dialogue on the integration of autonomous systems into our daily lives and emphasizes the importance of robust and reliable perception in shaping a safer and more efficient future.

Chapter 8

Conclusions and perspectives

In this thesis, we have embarked on a journey to explore the techniques of multi-sensor fusion for autonomous resilient perception, leveraging the power of classical methodologies and deep learning techniques. Our aim was to enhance the reliability and robustness of perception systems by exploiting the strengths of both approaches. By integrating classical methodologies such as the Extended Kalman Filter (EKF) with cutting-edge deep learning algorithms, we sought to achieve a comprehensive framework capable of effectively fusing sensor data and detecting anomalies or faults in measurements. Throughout this research, we delved into the fundamental concepts and principles of sensor fusion, recognizing its critical role in enabling autonomous systems to make accurate decisions in real-world environments. We explored the limitations and challenges associated with traditional sensor fusion techniques and acknowledged the potential of deep learning to overcome these hurdles. One of the key contributions of this thesis was the investigation of classical methodologies for sensor fusion, particularly the EKF. We examined the mathematical foundations of the EKF and its applicability in fusing measurements from multiple sensors. By incorporating this approach into our framework, we were able to exploit the temporal dependencies between sensor data and obtain more accurate and robust estimates of the environment. Moreover, we recognized the power of deep learning techniques in handling complex and unstructured data. Deep learning has revolutionized many fields, including computer vision, natural language processing, and speech recognition. We harnessed the potential of deep learning algorithms to enhance our perception system by employing them to detect outliers in measurements and identify sensor faults. The integration of deep learning methodologies enabled us to augment the resilience and adaptability of our framework, ensuring reliable and consistent performance even in the presence of sensor anomalies.

Throughout our experiments and evaluations, we witnessed the effectiveness and potential of our proposed multi-sensor fusion framework. By combining classical methodologies with

deep learning techniques, we achieved significant improvements in the perception capabilities of autonomous systems. Our framework demonstrated the ability to accurately estimate the state of the environment, detect outliers and faults in sensor data, and adapt to changing conditions. These advancements contribute towards the realization of autonomous systems that can operate reliably and effectively in dynamic and uncertain environments. However, it is important to acknowledge the limitations and areas for future research in this field. While our framework showed promising results, there are still challenges to address. The computational complexity of deep learning algorithms, especially in real-time applications, can be a bottleneck. Future work should focus on optimizing these algorithms and exploring more efficient architectures to enable their seamless integration into real-time perception systems.

Furthermore, the integration of classical methodologies and deep learning techniques opens up new avenues for research. Investigating novel ways to combine these approaches, such as incorporating deep learning into the EKF framework or designing hybrid fusion architectures, could yield even more robust and resilient perception systems. In conclusion, this thesis has contributed to the advancement of multi-sensor fusion for autonomous resilient perception. By leveraging classical methodologies such as the EKF and integrating deep learning techniques for outlier detection and sensor fault identification, we have built a comprehensive framework capable of fusing sensor data effectively. Our experimental results showcase the enhanced reliability and robustness of our proposed framework, paving the way for the development of autonomous systems that can operate autonomously in complex and uncertain environments.

As we look to the future, the fusion of classical methodologies and deep learning techniques will continue to play a pivotal role in the evolution of autonomous systems. With ongoing advancements in sensor technology, the proliferation of deep learning algorithms, and the increasing demand for resilient perception, this field holds great potential for further exploration and innovation. By combining the strengths of classical methodologies and deep learning techniques, we can unlock the full potential of multi-sensor fusion and create perception systems that exhibit unparalleled reliability, adaptability, and autonomy.

In the following sections we describe the perspectives for each of the aspects and topics that have been covered along the dissertation.

8.1 Perspectives for sensor resilience for localization and SLAM

This section explores the future perspectives for sensor resilience in the domains of localization and Simultaneous Localization and Mapping (SLAM) within the context of the dissertation. The thesis made significant contributions by leveraging classical methodologies, including the Extended Kalman Filter (EKF), and incorporating deep learning techniques for outlier detection in measurements. Building upon these accomplishments, this section delves into potential avenues of research and development for enhancing sensor resilience in the context of localization and SLAM.

Future advancements in sensor technology hold immense potential for improving sensor resilience in localization and SLAM. Emerging sensor modalities, such as LiDAR, radar, and depth cameras, are capable of providing richer and more comprehensive data. Integration of these modalities with existing sensor suites can enhance the robustness and reliability of the perception system. Exploring the benefits of multi-modal sensor fusion and investigating novel sensor technologies will contribute to further advancements in sensor resilience. Furthermore, the integration of classical methodologies, such as the EKF, with deep learning techniques opens up exciting possibilities for sensor resilience in localization and SLAM. Future research should focus on seamlessly combining these approaches to maximize their complementary strengths. Investigating novel fusion architectures that leverage both classical methodologies and deep learning algorithms can enhance the resilience and accuracy of localization and mapping processes. The development of hybrid approaches that fuse the temporal dependencies captured by classical methods with the representation learning capabilities of deep learning can lead to more robust and reliable sensor fusion frameworks. Then, ensuring the robustness of sensor measurements against faults and anomalies is crucial for resilient localization and SLAM. Future research should concentrate on advancing sensor fault detection and mitigation techniques. Expanding on the work done in the thesis, further exploration of deep learning algorithms for real-time detection of sensor anomalies and faulty measurements can enhance the ability to detect and mitigate sensor faults. Investigating adaptive fusion algorithms that dynamically select and weight sensor measurements based on their reliability and confidence levels will contribute to the development of more resilient sensor fusion frameworks.

The integration of machine learning algorithms within the realm of localization and SLAM offers exciting possibilities for improving sensor resilience. Future research should focus on developing learning-based approaches that reduce dependency on specific sensor modalities and enhance adaptability to different environments. Investigating deep learning

techniques for feature extraction and mapping in SLAM can enhance the ability to handle complex and dynamic environments. Furthermore, exploring reinforcement learning methods to optimize sensor selection and fusion strategies can significantly improve localization accuracy and sensor resilience. Furthermore, robust uncertainty quantification and management are vital for sensor resilience in localization and SLAM. Future research should concentrate on developing techniques for estimating and propagating uncertainties in fused sensor measurements, accounting for both systematic and random errors. Investigating adaptive sensor fusion algorithms that dynamically adjust their behavior based on confidence levels and uncertainties associated with each sensor will enhance the robustness and reliability of the localization and mapping processes.

Validation and evaluation of the proposed techniques in real-world scenarios and diverse environments are essential for the advancement of sensor resilience in localization and SLAM. Future research should focus on conducting experiments in challenging conditions, such as adverse weather, dynamic environments, or sensor failures, to assess the robustness and resilience of the sensor fusion framework. Collaborating with industry partners and autonomous vehicle manufacturers to deploy and validate the developed techniques in commercial applications will bridge the gap between academic research and practical implementation.

8.2 Perspectives for improvement of sensing abilities

This section explores future perspectives for the improvement of sensing abilities within the context of the dissertation. The thesis made significant contributions by employing various visual odometry systems applied to Unmanned Aerial Vehicles (UAVs). Building upon these achievements, this section discusses potential avenues of research and development for enhancing the sensing capabilities of autonomous systems.

Advancements in sensor technology play a vital role in improving the sensing abilities of autonomous systems. Future research should focus on exploring emerging sensor technologies and their integration with existing sensor suites. Sensors such as improved cameras, LiDAR, and thermal sensors can enhance the perception capabilities of UAVs, enabling them to operate in diverse environmental conditions. Additionally, investigating miniaturized and lightweight sensor solutions can optimize the trade-off between sensing capabilities and the overall weight and energy consumption of the UAV system. The integration of multi-sensor fusion techniques is key to improving the robustness and reliability of sensing abilities. Future research should concentrate on developing advanced fusion algorithms that effectively combine data from various sensors, including visual odometry systems, inertial measurement

units, and environmental sensors. Investigating novel fusion architectures that leverage the strengths of classical methodologies, such as Kalman filters, and deep learning techniques can enhance the accuracy and resilience of sensor fusion. Accurate sensor calibration and synchronization are critical for achieving precise and reliable sensing abilities. Future research should focus on developing techniques for calibrating and synchronizing sensors, particularly in UAV platforms where size, weight, and power constraints pose challenges. Investigating self-calibration and online calibration methods can reduce the reliance on manual calibration procedures, enabling real-time adaptability and robustness in the face of sensor drift and changing environmental conditions. Furthermore, autonomous systems often operate in challenging environments with limited visibility, adverse weather conditions, and dynamic surroundings. Future research should concentrate on enhancing the sensing abilities of UAVs in such challenging scenarios. Investigating sensor fusion techniques that can effectively handle occlusions, sensor noise, and environmental disturbances can improve perception accuracy. Additionally, exploring advanced imaging techniques, such as thermal imaging or hyperspectral imaging, can enable the detection and recognition of objects in low-light conditions or in the presence of camouflage.

Finally, efficient and real-time processing of sensor data is essential for the timely decision-making capabilities of autonomous systems. Future research should focus on developing algorithms and architectures that optimize the computational efficiency of sensor fusion and perception tasks. Exploring techniques such as hardware acceleration, distributed computing, and task allocation can ensure real-time performance while optimizing resource utilization in UAV systems. Validation and benchmarking of sensing abilities are crucial for assessing system performance and advancing the field. Future research should concentrate on developing standardized datasets, metrics, and evaluation protocols for comparing and benchmarking different sensing algorithms and techniques. Collaborative efforts within the research community and industry partnerships can facilitate the collection of comprehensive datasets in various operating conditions, enabling rigorous evaluation and validation of sensing abilities.

8.3 Perspectives for sensor data processing and generation

This section explores future perspectives for sensor data processing and generation within the context of the dissertation. The thesis made significant contributions by employing classical methodologies to generate sensor data and utilizing deep generative learning techniques to generate realistic sensor data. Building upon these accomplishments, this section discusses

potential avenues of research and development for enhancing the processing and generation of sensor data in autonomous systems.

Advancements in sensor data processing techniques play a crucial role in improving the quality, accuracy, and efficiency of sensor data utilized in autonomous systems. Future research should focus on exploring novel algorithms and methodologies for processing sensor data from diverse sources, including visual, inertial, and environmental sensors. Investigating advanced signal processing techniques, such as filtering, denoising, and feature extraction, can enhance the robustness and reliability of sensor data. Additionally, developing efficient and real-time data processing algorithms that can handle the high volume of sensor data generated by autonomous systems will be crucial for their practical implementation. The integration of classical methodologies and deep learning approaches offers promising possibilities for enhancing sensor data processing. Future research should concentrate on seamlessly combining the strengths of classical methodologies, such as statistical signal processing and model-based approaches, with deep learning techniques, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Investigating hybrid fusion architectures that leverage the complementary strengths of these approaches can enhance the accuracy and resilience of sensor data processing. Furthermore, developing novel algorithms that incorporate prior knowledge and domain-specific constraints within deep learning frameworks can improve the interpretability and reliability of generated sensor data. Deep generative learning techniques have demonstrated potential in generating realistic sensor data, which can be valuable for various applications, including training and testing autonomous systems. Future research should focus on developing advanced generative models, such as generative adversarial networks (GANs) and variational autoencoders (VAEs), tailored specifically for sensor data generation. Investigating techniques for incorporating domain knowledge, physical constraints, and uncertainty estimation within generative models can enhance their fidelity and applicability to real-world scenarios. Additionally, exploring techniques for generating diverse and representative sensor data that encompass various environmental conditions, sensor characteristics, and anomalies will be crucial for improving the generalization and resilience of autonomous systems. Data augmentation and simulation techniques can significantly contribute to the improvement of sensor data processing and generation. Future research should concentrate on developing advanced data augmentation algorithms that can effectively expand the diversity and quantity of training data. Investigating simulation environments that accurately model sensor characteristics, environmental conditions, and system dynamics can enable the generation of synthetic sensor data for training and testing autonomous systems. Furthermore, developing techniques for

bridging the gap between synthetic and real sensor data can enhance the transferability and applicability of simulated data to real-world scenarios.

Finally, quantifying and managing uncertainty in sensor data is crucial for ensuring reliable and robust decision-making in autonomous systems. Future research should focus on developing techniques for uncertainty quantification in sensor data processing and generation. Investigating methods for estimating and propagating uncertainties within sensor data fusion frameworks can enhance the reliability and resilience of autonomous systems. Additionally, exploring techniques for representing and incorporating uncertainties within generative models can improve the trustworthiness and interpretability of generated sensor data. Validation and benchmarking of sensor data processing and generation techniques are essential for assessing their performance and advancing the field. Future research should concentrate on developing standardized evaluation protocols, metrics, and benchmark datasets that enable rigorous comparison between the generated data and the real data from sensors.

References

[1] Francesco Martinelli and Fabrizio Romanelli. A slam algorithm based on range and bearing estimation of passive uhf-rfid tags. In *2021 IEEE International Conference on RFID Technology and Applications (RFID-TA)*, pages 20–23, 2021.

[2] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. A localization system for autonomous vehicles based on trilateration tags. In *2022 16th European Conference on Antennas and Propagation (EuCAP)*, pages 1–5, 2022.

[3] Francesco Martinelli and Fabrizio Romanelli. A resilient ro-slam algorithm with bearing reconstruction of detected landmarks. In *2022 3rd International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, pages 61–66, 2022.

[4] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Robust simultaneous localization and mapping using the relative pose estimation of trilateration uhf rfid tags. *IEEE Journal of Radio Frequency Identification*, 6:583–592, 2022.

[5] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. The role of the rfid polarization mismatch in the simultaneous localization and mapping problem. *IEEE Journal of Radio Frequency Identification*, 2023.

[6] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Exploiting the orientation of trilateration uhf rfid tags in robot localization and mapping. In *2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA)*, pages 5–8, 2022.

[7] Fabrizio Romanelli, Francesco Martinelli, and Emidio Di Giampaolo. Robust simultaneous localization and mapping using range and bearing estimation of radio ultra high frequency identification tags. *IEEE Transactions on Control Systems Technology*, 31(2):772–785, 2023.

[8] Francesco Martinelli, Simone Mattogno, and Fabrizio Romanelli. A resilient solution to range-only slam based on a decoupled landmark range and bearing reconstruction. *Robotics and Autonomous Systems*, 160:104324, 2023.

[9] Fabrizio Romanelli, Francesco Martinelli, and Emidio Di Giampaolo. Robust simultaneous localization and mapping using range and bearing estimation of radio ultra high frequency identification tags. In *2023 IEEE Conference on Control Technology and Applications (CCTA)*, pages 305–310. IEEE, 2023.

[10] Fabrizio Romanelli, Francesco Martinelli, and Simone Mattogno. Resilient simultaneous localization and mapping fusing ultra wide band range measurements and visual odometry. *Journal of Intelligent & Robotic Systems*, 109(64), 2023.

[11] Lorenzo Bianchi, Daniele Carnevale, Roberto Masocco, Simone Mattogno, Federico Oliva, Fabrizio Romanelli, and Alessandro Tenaglia. Efficient visual sensor fusion for autonomous agents. In *7th International Conference on Control, Automation and Diagnosis (ICCAD'23)*. IEEE, 2023.

[12] Lorenzo Bianchi, Daniele Carnevale, Fabio Del Frate, Roberto Masocco, Simone Mattogno, Fabrizio Romanelli, and Alessandro Tenaglia. A novel distributed architecture for unmanned aircraft systems based on robot operating system 2. *IET Cyber-Systems and Robotics*, 5(1):e12083, 2023.

[13] Alessandro Navone, Fabrizio Romanelli, Marco Ambrosio, Mauro Martini, Simone Angarano, and Marcello Chiaberge. Lavender autonomous navigation with semantic segmentation at the edge. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2023.

[14] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Exploiting polarization mismatch to estimate the orientation of rotating uhf rfid tags. *IEEE Journal of Radio Frequency Identification*, 2023.

[15] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Considering polarization mismatch in modeling the rfid phase offset variability for tag localization. In *2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA)*, pages 21–24, 2022.

[16] Emidio Di Giampaolo, Francesco Martinelli, and Fabrizio Romanelli. Exploiting the electromagnetic coupling to estimate the close motion of uhf-rfid tagged objects. In *2023 IEEE 13th International Conference on RFID Technology and Applications (RFID-TA)*, 2023.

[17] Fabrizio Romanelli and Francesco Martinelli. Synthetic sensor data generation exploiting deep learning techniques and multimodal information. *IEEE Sensors Letters*, 7(7):1–4, 2023.

[18] Fabrizio Romanelli and Francesco Martinelli. Synthetic sensor measurement generation with noise learning and multi-modal information. *IEEE Access*, 11:111765–111788, 2023.

[19] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007.

[20] H. B. Mitchell. *Multi-sensor data fusion an introduction*. Springer Verlag, Berlin; New York, 2007.

[21] D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[22] D. Dubois and H. Prade. *Théorie des possibilités: applications à la représentation des connaissances en informatique*. Méthode + programmes. Masson, 1985.

[23] Alberto Elfes. Occupancy grids: a probabilistic framework for robot perception and navigation. In *Carnegie Mellon University ProQuest Dissertations Publishing*, 1989.

[24] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University-Computer and Information Sciences*, 34(8):6019–6039, 2022.

[25] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[26] Ramazan Havangi. Mobile robot localization based on pso estimator. *Asian Journal of Control*, 21(4):2167–2178, 2019.

[27] Hossein Vahid Dastjerdi, Mohammad Bagher Menhaj, and Saeideh Shataei. Self-localization of humanoid robots using particle swarm optimization algorithm. In *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pages 98–102. IEEE, 2016.

[28] Haijiang Wang, Wenhong Liu, Fugui Zhang, Simon X Yang, and Lin Zhang. A ga-fuzzy logic based extended kalman filter for mobile robot localization. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 319–323. IEEE, 2015.

[29] Shreedhar D Rajurkar, Aniket K Kar, Subhajit Goswami, and Nishchal K Verma. Optimal path estimation and tracking for an automated vehicle using ga optimized fuzzy controller. In *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pages 365–370. IEEE, 2016.

[30] Ali R Vahdat, Naser NourAshrafoddin, and Saeed Shiry Ghidary. Mobile robot global localization using differential evolution and particle swarm optimization. In *2007 IEEE congress on evolutionary computation*, pages 1527–1534. IEEE, 2007.

[31] Junfei Huo, Liling Ma, Yuanlong Yu, and Junzheng Wang. Hybrid algorithm based mobile robot localization using de and pso. In *Proceedings of the 32nd Chinese Control Conference*, pages 5955–5959. IEEE, 2013.

[32] Ramazan Havangi, Mohammad Ali Nekoui, and Mohammad Teshnehlab. A multi swarm particle filter for mobile robot localization. *International Journal of Computer Science*, 7(3):15–22, 2010.

[33] Andry M Pinto, António P Moreira, and Paulo G Costa. A localization method based on map-matching and particle swarm optimization. *Journal of Intelligent & Robotic Systems*, 77:313–326, 2015.

[34] Sabita Mali. Mobile robot localization using multi-objective optimization. *Int. J. Latest Technol. Eng., Manag. Appl. Sci.-IJLTEMAS*, 4:21–25, 2015.

[35] Feng Lu and Evangelos E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[36] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 1442–1447 vol.3, 1991.

[37] Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Moving objects detection by conflict analysis in evidential grids. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1122–1127, 2011.

[38] Dizan Vasquez, Fabrizio Romanelli, Thierry Fraichard, and Christian Laugier. Fast object extraction from bayesian occupancy grids using self organizing networks. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6, 2006.

[39] Trung-Dung Vu. *Vehicle Perception: Localization, Mapping with Detection, Classification and Tracking of Moving Objects*. Theses, Institut National Polytechnique de Grenoble - INPG, September 2009.

[40] Randall Smith, Matthew Self, and Peter Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics*, pages 167–193. Springer New York, New York, NY, 1990.

[41] Sebastian Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21, 07 2000.

[42] Qadeer Baig. Multi sensor data fusion for detection and tracking of moving objects from a dynamic autonomous vehicle. *HAL*, 2012, 2012.

[43] Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Credibilist occupancy grids for vehicle perception in dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 84–89. IEEE, 2011.

[44] Daniel Pagac, Eduardo Mario Nebot, and Hugh Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 14(4):623–629, 1998.

[45] Glenn Shafer. Dempster-shafer theory. *Encyclopedia of artificial intelligence*, 1:330–331, 1992.

[46] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic systems*, 18:249–275, 1997.

[47] David M Cole and Paul M Newman. Using laser range data for 3d slam in outdoor environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1556–1563. IEEE, 2006.

[48] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[49] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11, 2017.

[50] Juraj Kabzan, Miguel I Valls, Victor JF Reijgwart, Hubertus FC Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *Journal of Field Robotics*, 37(7):1267–1294, 2020.

[51] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[52] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

[53] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[54] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015.

[55] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*, 43:55–81, 2015.

[56] João Pedro Mucheroni Covolan, Antonio Carlos Sementille, and Silvio Ricardo Rodrigues Sanches. A mapping of visual slam algorithms and their applications in augmented reality. In *2020 22nd Symposium on Virtual and Augmented Reality (SVR)*, pages 20–29. IEEE, 2020.

[57] Myriam Servières, Valérie Renaudin, Alexis Dupuis, and Nicolas Antigny. Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking. *Journal of Sensors*, 2021:1–26, 2021.

[58] Jianjun Gui, Dongbing Gu, Sen Wang, and Huosheng Hu. A review of visual inertial odometry from filtering and optimisation perspectives. *Advanced Robotics*, 29(20):1289–1301, 2015.

[59] Chang Chen, Hua Zhu, Menggang Li, and Shaoze You. A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives. *Robotics*, 7(3):45, 2018.

[60] Guoquan Huang. Visual-inertial navigation: A concise review. In *2019 international conference on robotics and automation (ICRA)*, pages 9572–9582. IEEE, 2019.

[61] Kang Chen, Yu-Kun Lai, and Shi-Min Hu. 3d indoor scene modeling from rgb-d data: a survey. *Computational Visual Media*, 1:267–278, 2015.

[62] Shishun Zhang, Longyu Zheng, and Wenbing Tao. Survey and evaluation of rgb-d slam. *IEEE Access*, 9:21367–21387, 2021.

[63] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1), 2022.

[64] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, 81:155–166, 2009.

[65] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[66] Konstantinos Boikos and Christos-Savvas Bouganis. Semi-dense slam on an fpga soc. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2016.

[67] Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In *2019 International conference on robotics and automation (ICRA)*, pages 5218–5223. IEEE, 2019.

[68] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*, 2016.

[69] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[70] Simone Bianco, Gianluigi Ciocca, and Davide Marelli. Evaluating the performance of structure from motion pipelines. *Journal of Imaging*, 4(8):98, 2018.

[71] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[72] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[73] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[74] Bruce Canovas, Michèle Rombaut, Amaury Nègre, Denis Pellerin, and Serge Olympieff. Speed and memory efficient dense rgb-d slam in dynamic scenes. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4996–5001. IEEE, 2020.

[75] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[76] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[77] Zongqian Zhan, Wenjie Jian, Yihui Li, and Yang Yue. A slam map restoration algorithm based on submaps and an undirected connected graph. *IEEE Access*, 9:12657–12674, 2021.

[78] Mohamed Abouzahir, Abdelhafid Elouardi, Rachid Latif, Samir Bouaziz, and Abdelouahed Tajer. Embedding slam algorithms: Has it come of age? *Robotics and Autonomous Systems*, 100:14–26, 2018.

[79] Jincheng Yu, Feng Gao, Jianfei Cao, Chao Yu, Zhaoliang Zhang, Zhengfeng Huang, Yu Wang, and Huazhong Yang. Cnn-based monocular decentralized slam on embedded fpga. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 66–73. IEEE, 2020.

[80] Fabrizio Romanelli. ORB-SLAM2 - New parameters management, ARM compilation, CUDA GPU compatibility, 8 2023.

[81] Qingwen Xu, Haofei Kuang, Laurent Kneip, and Sören Schwertfeger. Rethinking the fourier-mellin transform: Multiple depths in the camera's view. *Remote Sensing*, 13(5):1000, 2021.

[82] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291, 2018.

[83] Ruihao Li, Sen Wang, and Dongbing Gu. Deepslam: A robust monocular slam system with unsupervised deep learning. *IEEE Transactions on Industrial Electronics*, 68(4):3577–3587, 2021.

[84] Rong Kang, Jieqi Shi, Xueming Li, Yang Liu, and Xiao Liu. Df-slam: A deep-learning enhanced visual slam system based on deep local features, 2019.

[85] ChaoQiang Zhao, QiYu Sun, ChongZhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627, jun 2020.

[86] Ruan Xiaogang, Yan Wenjing, Huang Jing, Guo Peiyuan, and Guo Wei. Monocular depth estimation based on deep learning:a survey. In *2020 Chinese Automation Congress (CAC)*, pages 2436–2440, 2020.

[87] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.

[88] Qadeer Baig, Olivier Aycard, Trung Dung Vu, and Thierry Fraichard. Fusion between laser and stereo vision data for moving objects tracking in intersection like scenario. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 362–367, 2011.

[89] Baerbel Grabe, Thorsten Ike, and Michael Hoetter. Evaluation method of grid based representation from sensor data. In *2009 IEEE Intelligent Vehicles Symposium*, pages 1245–1250, 2009.

[90] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, CIVR '07, page 401–408, New York, NY, USA, 2007. Association for Computing Machinery.

[91] Asma Azim and Olivier Aycard. Detection, classification and tracking of moving objects in a 3d environment. In *2012 IEEE Intelligent Vehicles Symposium*, pages 802–807, 2012.

[92] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[93] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[94] Song-Chun Zhu, Rong Zhang, and Zhuowen Tu. Integrating bottom-up/top-down for object recognition by data driven markov chain monte carlo. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 738–745 vol.1, 2000.

[95] Bartłomiej Jacek Kubica. *Interval Calculus*, pages 5–16. Springer International Publishing, Cham, 2019.

[96] H.-J. Zimmermann. Fuzzy set theory. *WIREs Computational Statistics*, 2(3):317–332, 2010.

[97] Thomas Burks, Warren Dixon, and Vijay Subramanian. Sensor fusion using fuzzy logic enhanced kalman filter for autonomous vehicle guidance in citrus groves. *American Society of Agricultural and Biological Engineers ISSN*, 52:1411–1422, 09 2009.

[98] A. P. Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(2):205–247, 1968.

[99] Philippe Smets and Robert Kennes. The transferable belief model. *Artificial Intelligence*, 66(2):191–234, 1994.

[100] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[101] P. Smets. The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):447–458, 1990.

[102] Ronald R. Yager. On the relationship of methods of aggregating evidence in expert systems. *Cybernetics and Systems*, 16(1):1–21, 1985.

[103] Simon Arvidsson, Marcus Gullstrand, Beril Sirmacek, and Maria Riveiro. Sensor fusion and convolutional neural networks for indoor occupancy prediction using multiple low-cost low-resolution heat sensor data. *Sensors*, 21(4):1036, 2021.

[104] Jingchang Li, Qi Zhou, Longchao Cao, Yanzhi Wang, and Jiexiang Hu. A convolutional neural network-based multi-sensor fusion approach for in-situ quality monitoring of selective laser melting. *Journal of Manufacturing Systems*, 64:429–442, 2022.

[105] Bohuai Xiao, Xiaolan Xie, and Chengyong Yang. Multi-sensor data fusion based on gcn-lstm. *Int. J. Innov. Comput. Inf. Control*, 18(5):1363–1382, 2022.

[106] Kriti Kumar, Saurabh Sahu, Angshul Majumdar, and M Girish Chandra. Autofuse: A semi-supervised autoencoder based multi-sensor fusion framework. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021.

[107] Sepehr Pashoutani, Jinying Zhu, Chungwook Sim, Kwanghee Won, Brian A Mazzeo, and W Spencer Guthrie. Multi-sensor data collection and fusion using autoencoders in condition evaluation of concrete bridge decks. *Journal of Infrastructure Preservation and Resilience*, 2(1):1–12, 2021.

[108] Siddharth Roheda, Hamid Krim, and Benjamin S Riggan. Robust multi-modal sensor fusion: An adversarial approach. *IEEE Sensors Journal*, 21(2):1885–1896, 2020.

[109] Sz-Rung Shiang, Anatole Gershman, and Jean Hyaejin Oh. A generalized model for multimodal perception. In *Proceedings of AAAI '17 Fall Symposium*, November 2017.

[110] Christophe Mollaret, Alhayat Ali Mekonnen, Frédéric Lerasle, Isabelle Ferrané, Julien Pinquier, Blandine Boudet, and Pierre Rumeau. A multi-modal perception based assistive robotic system for the elderly. *Computer Vision and Image Understanding*, 149:78–97, 2016.

[111] Huaping Liu, Fuchun Sun, and Xinyu Zhang. Robotic material perception using active multimodal fusion. *IEEE Transactions on Industrial Electronics*, 66(12):9878–9886, 2018.

[112] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Outlier detection: A survey. *ACM Computing Surveys*, 14:15, 2007.

[113] K Chen, SC Lu, and HS Teng. Adaptive real-time anomaly detection using inductively generated sequential patterns,". In *Fifth Intrusion Detection Workshop, SRI International, Menlo Park, CA*, 1990.

[114] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005.

[115] Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[116] Markos Markou and Sameer Singh. Novelty detection: a review—part 2:: neural network based approaches. *Signal processing*, 83(12):2499–2521, 2003.

[117] Rob Saunders and JOHN S Gero. The importance of being emergent. In *Proceedings of Artificial Intelligence in Design*, 2000.

[118] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[119] Jonathan Francis Dale Addison, Stefan Wermter, and John MacIntyre. Effectiveness of feature extraction in neural network architectures for novelty detection. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470)*, volume 2, pages 976–981. IET, 1999.

[120] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, Pang-Ning Tan, Vipin Kumar, Jaideep Srivastava, and Paul Dokas. Minds-minnesota intrusion detection system. *Next generation data mining*, pages 199–218, 2004.

[121] Bovas Abraham and George EP Box. Bayesian analysis of some outlier problems in time series. *Biometrika*, 66(2):229–236, 1979.

[122] Bovas Abraham and Alice Chuang. Outlier detection and time series modeling. *Technometrics*, 31(2):241–248, 1989.

[123] Tom Michael Mitchell et al. *Machine learning*, volume 1. McGraw-hill New York, 2007.

[124] Stephan R Sain. The nature of statistical learning theory, 1996.

[125] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509, 2006.

[126] Dipankar Dasgupta and Nivedita Sumi Majumdar. Anomaly detection in multidimensional data using negative selection algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1039–1044. IEEE, 2002.

[127] Dipankar Dasgupta and Fernando Nino. A comparison of negative and positive selection algorithms in novel pattern detection. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 1, pages 125–130. IEEE, 2000.

[128] Eamonn Keogh, Stefano Lonardi, and Bill'Yuan-chi' Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 550–556, 2002.

[129] Jaideep Vaidya and Chris Clifton. Privacy-preserving outlier detection. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 233–240. IEEE, 2004.

[130] Korbinian Hagn and Oliver Grau. Improved sensor model for realistic synthetic data generation. In *Proceedings of the 5th ACM Computer Science in Cars Symposium*, CSCS '21, New York, NY, USA, 2021. Association for Computing Machinery.

[131] Zhihong Zhang, Jinli Suo, and Qionghai Dai. Denoising of event-based sensors with deep neural networks. In *Optoelectronic Imaging and Multimedia Technology VIII*, volume 11897, pages 203–209. SPIE, 2021.

[132] Ebtesam Almazrouei, Gabriele Gianini, Nawaf Almoosa, and Ernesto Damiani. A deep learning approach to radio signal denoising. In *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*, pages 1–8, 2019.

[133] Moustafa Farid Alzantot, Supriyo Chakraborty, and Mani B. Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 188–193, 2017.

[134] Skyler Norgaard, Ramyar Saeedi, Keyvan Sasani, and Assefaw H. Gebremedhin. Synthetic sensor data generation for health applications: A supervised deep learning approach. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1164–1167, 2018.

[135] Korbinian Hagn and Oliver Grau. Improved sensor model for realistic synthetic data generation. In *Proceedings of the 5th ACM Computer Science in Cars Symposium*, pages 1–9, 2021.

[136] Lars Ruthotto and Eldad Haber. An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021.

[137] Ali Jahanian, Xavier Puig, Yonglong Tian, and Phillip Isola. Generative models as a data source for multiview representation learning. *arXiv preprint arXiv:2106.05258*, 2021.

[138] Jakub M Tomczak. *Deep generative modeling*. Springer, 2022.

[139] Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021.

[140] Gang Hua and Dongdong Chen. Chapter 5 - deep conditional image generation: Towards controllable visual pattern modeling. In E.R. Davies and Matthew A. Turk, editors, *Advanced Methods and Deep Learning in Computer Vision*, Computer Vision and Pattern Recognition, pages 191–219. Academic Press, 2022.

[141] Robert Giaquinto and Arindam Banerjee. Gradient boosted normalizing flows. *Advances in Neural Information Processing Systems*, 33:22104–22117, 2020.

[142] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

[143] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[144] Ho Hin Lee, Yucheng Tang, Shunxing Bao, Yan Xu, Qi Yang, Xin Yu, Agnes B. Fogo, Raymond Harris, Mark P. de Caestecker, Jeffery M. Spraggins, Mattias Heinrich, Yuankai Huo, and Bennett A. Landman. Supervised deep generation of high-resolution arterial phase computed tomography kidney substructure atlas. In Olivier Colliot and Ivana Išgum, editors, *Medical Imaging 2022: Image Processing*, volume 12032, page 120322S. International Society for Optics and Photonics, SPIE, 2022.

[145] Yongjie Shi, Xianghua Ying, and Jinfa Yang. Deep unsupervised domain adaptation with time series sensor data: A survey. *Sensors*, 22(15), 2022.

[146] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[147] Sohel J. Patel and Maciej J. Zawodniok. 3d localization of rfid antenna tags using convolutional neural networks. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022.

[148] Amira Mimouna, Anouar Ben Khalifa, Ihsen Alouani, Abdelmalik Taleb-Ahmed, Atika Rivenq-Menhaj, and Najoua Ben Amara. Lstm-based system for multiple obstacle detection using ultra-wide band radar. *13th International Conference on Agents and Artificial Intelligence, ICAART*, 2021.

[149] L. Ramos and J. Subramanyam. Maverick* research: Forget about your real data - synthetic data is the future of ai. *Gartner Research*, 2021.

[150] Emidio Di Giampaolo and Francesco Martinelli. Range and bearing estimation of an uhf-rfid tag using the phase of the backscattered signal. *IEEE Journal of Radio Frequency Identification*, 4(4):332–342, 2020.

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

[152] Chuanyang Wang, Houzeng Han, Jian Wang, Hang Yu, and Deng Yang. A robust extended kalman filter applied to ultrawideband positioning. *Mathematical Problems in Engineering*, 2020:1–12, 2020.

[153] John Maindonald. Modern multivariate statistical techniques: Regression, classification and manifold learning. *Journal of Statistical Software*, 29:1–3, 2009.

[154] Guobin Chang. Robust kalman filtering based on mahalanobis distance as outlier judging criterion. *Journal of Geodesy*, 88(4):391–401, 2014.

[155] Jian-Guo Wang. Reliability analysis in kalman filtering. *Journal of Global Positioning Systems*, 8(1):101–111, 2009.

[156] Rüdiger Lehmann. Improved critical values for extreme normalized and studentized residuals in gauss–markov models. *Journal of geodesy*, 86:1137–1146, 2012.

[157] Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Asynchronous multi-sensor fusion for 3d mapping and localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5994–5999. IEEE, 2018.

[158] Sebastian Villar, Sebastian Torcida, and Gerardo Acosta. Median filtering: A new insight. *Journal of Mathematical Imaging and Vision*, 58:1–17, 05 2017.

[159] Ginu George, Rinoy Mathew Oommen, Shani Shelly, Stephie Sara Philipose, and Ann Mary Varghese. A survey on various median filtering techniques for removal of impulse noise from digital image. In *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, pages 235–238, 2018.

[160] Anwar Shah, Javed Iqbal Bangash, Abdul Waheed Khan, Imran Ahmed, Abdullah Khan, Asfandyar Khan, and Arshad Khan. Comparative analysis of median filter and its variants for removal of impulse noise from gray scale images. *Journal of King Saud University - Computer and Information Sciences*, 34(3):505–519, 2022.

[161] C. A. R. Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 01 1962.

[162] Emidio Di Giampaolo and Fernando Bardati. A projective approach to electromagnetic propagation in complex environments. *Progress In Electromagnetics Research B*, 13:357–383, 2009.

[163] E Di Giampaolo and Fernando Bardati. Analytical model of multiple wedge-diffracted ray congruence. *Electromagnetics*, 23(6):509–523, 2003.

[164] E Di Giampaolo, M Sabbadini, and F Bardati. Astigmatic beam tracing for gtd/utd methods in 3-d complex environments. *Journal of Electromagnetic Waves and Applications*, 15(4):439–460, 2001.

[165] Francesco Martinelli. Simultaneous localization and mapping using the phase of passive uhf-rfid signals. *Journal of Intelligent & Robotic Systems*, 94:711–725, 2019.

[166] HJ Li, SH Tang, and J Huang. Discussion for the selection of constant in selecting weight iteration method in robust estimation. *Science of Surveying and Mapping*, 31(6):70–72, 2006.

[167] Emidio Di Giampaolo and Francesco Martinelli. Mobile robot localization using the phase of passive uhf rfid signals. *IEEE Transactions on Industrial Electronics*, 61(1):365–376, 2013.

[168] Yongtao Ma, Hankai Liu, Yunlei Zhang, and Yue Jiang. The influence of the nonideal phase offset on sar-based localization in passive uhf rfid. *IEEE Transactions on Antennas and Propagation*, 68(8):6346–6354, 2020.

[169] Fabrizio Romanelli. ORB-SLAM2 - New parameters management, ARM compilation, CUDA GPU compatibility. *Version 1.0.0*, 8 2021.

[170] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35, 01 2016.

[171] Y. Cao and G. Beltrame. Vir-slam: visual, inertial, and ranging slam for single and multi-robot systems. *Autonomous Robots*, 45:905–917, 2021.

[172] L. Alzubaidi, J. Zhang, and A.J. et al. Humaidi. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 53(8), 2021.

[173] David Prokhorov, Dmitry Zhukov, Olga Barinova, Konushin Anton, and Anna Vorontsova. Measuring robustness of visual slam. In *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6, 2019.

[174] A. Viana-Lora and M. G. Nel-lo Andreu. Approaching the social impact of research through a literature review. *International Journal of Qualitative Methods*, 1(20), 2021.

[175] Lutz Bornmann. What is societal impact of research and how can it be assessed? a literature survey. *Journal of the American Society for Information Science and Technology*, 64(2), 2012.

[176] H.F. Voigt. Social implications of technology: Introductory words. *Health and Technology*, 6, 2016.

[177] Caixia Mao, Ryu Koide, Alexander Brem, and Lewis Akenji. Technology foresight for social good: Social implications of technological innovation by 2050 from a global expert survey. *Technological Forecasting and Social Change*, 153:119914, 2020.