UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXVII Ciclo

UWB TECHNOLOGY FOR INDOOR LOCALIZATION: FROM THEORY TO PRACTICE

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Gianluigi Ferrari

Dottoranda: *Stefania Monica*

Gennaio 2015

Sbuca da sotto la neve
l’acqua del fossato.
Presto
lì
sull’argine fresco
nasceranno viole.
Ricordati di questo
nei giorni
senza profumo.

– Silvio Monica

Contents

List of Figures

List of Tables

List of Listings

Introduction

In recent years, the increasing importance of context- and location-dependent information has led to a growing interest in location-based applications and services. Among many interesting scenarios, accurate localization and real-time tracking of objects in indoor environments have many applications in various areas, such as: monitoring of people and goods in airports, subways, high security areas, and shopping malls; monitoring of patients in hospitals; and locating people and vehicles in industrial warehouses.

While the problem of outdoor localization is usually solved by the Global Positioning System (GPS) or other more accurate techniques, indoor localization is still a challenge. GPS is useless in indoor environments, since the signals from satellites cannot penetrate through walls and its typical accuracy is too poor for indoor localization applications. Moreover, the accurate localization of targets in indoor environments is particularly difficult because signal propagation inside buildings is affected by various phenomena, such as non-line-of-sight and multipath, caused by walls and obstacles. This dissertation focuses on the study and design of indoor localization techniques which are aimed at locating targets inside buildings. Throughout the dissertation, Ultra Wide Band (UWB) signaling is considered. In general terms, all scenarios involve a few nodes with known positions, denoted as Anchor Nodes (ANs), which properly cooperate, through collaborative signal processing, to estimate the (unknown) positions of target nodes (each equipped with a UWB sensor).

In particular, in Chapter 1 the localization problem is introduced and the UWB technology is described. In Chapter 2, the impact of ANs positioning on the local-

ization accuracy of a moving target in a specific scenario is investigated. In Chapter 3, a comparison between the performances of geometric and optimization-based approaches to the localization of static targets is performed. While the results presented in Chapters 2 and 3 are obtained through analytical framework and computer simulations, in Chapter 4 experimental results obtained with the PulsON 410 Ranging and Communications Modules (RCMs) produced by the company Time Domain (`www.timedomain.com`) are presented. This experimental investigation is expedient to derive a novel statistical model for range estimates between pairs of UWB sensors, which is then applied to improve the performance of few illustrative localization algorithms in various scenarios. In Chapter 5, a realistic scenario for indoor localization is presented and two efficient localization algorithms are investigated with PulsON 410 RCMs. Finally, a summary of the results is presented and conclusions are drawn.

Chapter 1

Survey of the Literature and Motivations

Nos esse quasi nanos
gigantium humeris insidentes

– Bernardus Carnotensis

1.1 Introduction

Wireless Sensor Networks (WSNs) allow collecting measures of physical quantities and processing them through smart sensors. The main feature of this type of sensors is that they can monitor environmental parameters and process the collected data locally and convey the obtained information to one or more central nodes [14]. Smart sensors are typically low-power devices which can measure a variety of mechanical, thermal, biological, chemical, optical, and magnetic properties of the environment. They typically consist of a processor (with limited computing power), a memory, and short-range transmitter/receiver.

Thanks to the availability of small, cheap, and smart sensors, the attention of the scientific community towards WSNs is increasing because they represent a leading choice to address some of the most interesting challenges, such as smart environments and the Internet of Things (IoT). A large number of applications can be addressed with WSNs, including: assisted living; security area surveillance; medical and biomedical monitoring; traffic control and safety; search and of victims in emergency situations; environmental monitoring; smart homes; localization and tracking of people, vechicles, and goods [1].

Wireless localization plays a key role among all the above interesting and challenging topics. As a matter of fact, in many applications the position of sensors must be known in order to make sure that their data is meaningful. Moreover, the position itself can be the data that needs to be monitored in many applications, for example in: military security [26]; home surveillance and assisted living [37]; medical supervision [9]; smart homes [56]; logistics [34]; and industrial monitoring [68, 70].

Localization systems can be categorized, depending on the target environment, as indoor or outdoor. For outdoor localization, Global Navigation Satellite Systems (GNSS) such as the Global Positioning System (GPS) can be used in a wide range of applications including tracking and asset management; transport navigation and guidance; synchronization of telecommunications networks; and geodetic survey. Unfortunately, GPS does not perform well in indoor environments, as the radio signal strength is too weak to penetrate through walls. Moreover, the accuracy of GPS localization is usually around 10 m and this is typically too poor for indoor localization, which often requires higher accuracy [16].

To address the inadequacy of GPS for the localization of targets inside buildings, proper indoor localization techniques and technologies are needed. Many application requirements (such as scalability, energy efficiency, cost, and required accuracy) influence the design of indoor localization systems [52]. Based on such requirements, the localization task can be solved by means of a variety of technologies based on inertial, optical, acoustic, and magnetic properties [16]. Moreover, different wireless technologies can be used to identify targets positions. Among them, Infrared Radiation (IR) localization systems are very common, since IR technology is available

in various devices, such as televisions, printers, and mobile phones. Moreover, IR devices are usually small and lightweight. However, IR-based localization systems have some limitations since they suffer from interference from fluorescent light and sunlight and they need Line-of-Sight (LoS) communication between transmitters and receivers. Moreover, the coverage range per infrastructure device is limited [20].

Radio Frequency (RF) technologies are commonly used in localization systems. For instance, Bluetooth-based localization systems can be considered [62]. The benefit of using Bluetooth for exchanging information between devices is that this technology guarantees a high security with low cost, low power, and small device size. Each Bluetooth tag has a unique identifier, which can be used for locating the Bluetooth tag. The accuracy of a Bluetooth-based localization system is typically between 2 m and 3 m, which may not be sufficient for some applications [20]. Another drawback of using Bluetooth technology for localization purposes is that a device discovery procedure needs to be run for each location update. This is the reason why Bluetooth devices have a latency unsuitable for real-time applications [27].

WiFi-based localization systems are commonly used, as they typically have a low cost, owing to the availability of a large number of WiFi devices in the majority of indoor environments [28]. Moreover, LoS between transmitter and receiver is not required in WiFi-based localization systems. Despite these good properties, WiFi-based localization systems also have some weaknesses. As a matter of fact, they can interfere with other devices operating on the same carrier, namely 2.4 GHz. Moreover, the accuracy of the final position estimates is typically greater than 1 m and it can considerably increase because of signal attenuation due to Non-Line-of-Sight (NLoS) and multipath, caused by walls and obstacles [20]. Such phenomena are the major sources of errors in indoor environments and a good localization system should be able to handle them.

To overcome the problems of the aforementioned technologies, Ultra Wide Band (UWB) technology is a good candidate for indoor localization systems, as it can theoretically reduce the impact of these sources of errors [35]. Localization systems based on UWB signaling offer various advantages, in terms of accuracy and scalability, over other localization technologies, as discussed in next section.

1.2 Ultra-Wide Band Technology

The acronym UWB was coined by the US Department of Defense in the late 1980s, and it became popular after the Federal Communications Commission (FCC) allowed the unlicensed use of UWB devices in February 2002 subject to emission constraints [57]. A UWB signal is characterized by very large bandwidth with respect to the conventional, also known as narrowband, systems. According to the FCC [57]:

- UWB signals with carrier frequency f_c higher than 2.5 GHz must have an absolute bandwidth larger than 500 MHz;
- UWB signals with f_c lower than 2.5 GHz must have a fractional bandwidth larger than 0.2.

The absolute bandwidth of a signal is calculated as the difference between the upper frequency f_H and the lower frequency f_L of the -10 dB emission points, namely $B = f_H - f_L$; the fractional bandwidth is defined as $B_{\text{frac}} = B/f_c$ [18].

In 2004, the IEEE established standardization group IEEE 802.15.4a aimed at defining a new physical layer for the already existing IEEE 802.15.4 standard for Wireless Personal Area Networks (WPANs). The new IEEE 802.15.4a standard, released in 2007, provides a physical layer for short-range low data rate communications and for high-precision ranging with low-power and low-cost devices, suitable for many applications [6].

According to the IEEE 802.15.4a standard, a UWB device can transmit in one or more of the following bands

- 250-750 MHz (sub-GHz)
- 3.244-4.742 GHz (low band)
- 5.944-10.234 GHz (high band)

and 16 channels are supported over such three bands [57].

The interest on UWB systems is growing fast because they represent a promising technology in many fields and some UWB localization devices are already available, produced by companies such as Decawave (`www.decawave.com`), Ubisense (`www.ubisense.net`), and Time Domain (`www.timedomain.com`).

One of the good properties which makes the UWB technology attractive in the context of WSNs is the low-power consumption which charachterizes UWB devices. As a matter of fact, since UWB signals occupy a very large portion in the spectrum and, at the same time, they need to coexist with other wireless technologies, it is necessary to impose low-power transmission for UWB systems in order to avoid interference problems with other devices operating in the same frequency spectrum. More precisely, the power spectral density must not exceed -41.3 dBm/MHz over the frequency band from 3.1 GHz to 10.6 GHz and must be even lower outside this band, depending on the specific application [18].

Due to their very large bandwidth, UWB systems transmit very short duration pulses, usually on the order of nanoseconds, with a low duty cycle. In other words, the ratio between the pulse duration and the average time between two consecutive pulses is usually kept small, leading to low energy consumption.

One of the most promising aspects of UWB signals is their potential for high-precision localization. As a matter of fact, short time duration pulses guarantee accurate Time of Flight (ToF) estimation of signals travelling between nodes. This implies that the distance between a transmitter and a receiver can be accurately determined, yielding high localization accuracy. At the opposite, pulses received via multiple paths in narrow band systems can easily overlap, causing wrong ToF estimates, hence wrong range estimates [72].

Furthermore, UWB signals are characterized by the capability of penetrating through obstacles, due to the large frequency spectrum that characterizes them, which includes both low- and high-frequency components [21]. Finally, it is worth observing that a UWB system can be implemented in baseband and this easies the design of transmitters and receivers, hence reducing the cost of the system [57].

Such unique aspects make the UWB technology a good candidate for accurate, low-cost, and low-power localization systems.

1.3 Localization Techniques

Many localization algorithms for WSNs have been proposed to provide per-node location information. With regard to the mechanisms used for location estimation, WSN localization approaches can be divided into two main categories: range-free and range-based [22]. Range-free approaches do not require the availability of range estimates between nodes. These techniques can be further divided into local techniques, such as the centroid algorithm [4] or the Approximate Point In Triangulation (APIT) algorithm [22], and hop-counting techniques, such as the Distance Vector Hop algorithm [51]. At the opposite, range-based approaches rely on the knowledge of inter-node ranges or angles.

Range-based localization techniques can be classified into active and passive [35]. In active techniques, all nodes are equipped with smart sensors which send information to a central localization system. Passive localization, instead, relies on the fact that wireless communications strongly depend on the surrounding environment. The scattering caused by small targets during signal propagation and the variance of measured signals can be used to detect and locate static targets or to track a moving targets [12].

In this dissertation, we focus on range-based algorithms with active tags. More precisely, we consider two-step localization techniques, where the first step consists in the estimation of signal parameters, such as the ToF, the Angle of Arrival (AoA), or the Received Signal Strength (RSS). The position of a target is then estimated in the second step, which relies on proper processing of the signal parameters estimated in the first step [46].

Range-estimation techniques based on AoA rely on the measurements of angles between nodes, which are usually taken by means of antenna arrays. However, the installation cost of antenna arrays can be high, thus reducing the advantage of low-cost UWB transceivers. Moreover, the number of paths of UWB signals may be very large, making accurate angle estimation very challenging, especially in indoor environments where the presence of many objects can cause relevant scattering [19]. For these reasons, the AoA approach is not suitable for UWB localization systems.

In the presence of a relation between the received power of a signal travelling between two nodes and their distance (as is the case for LoS transmissions), the latter can be estimated from the RSS measurement at the receiver node assuming that the transmitted signal power is known. The path loss during propagation is characterized by the following Friis formula (in the logarithmic domain) [18]

$$\bar{P}(d) = P_0 - 10\beta \log_{10} \frac{d}{d_0} \tag{1.1}$$

where: P_0 is the (known) power at the reference distance d_0; β is the path loss exponent; and $\bar{P}(d)$ is the average received power at the distance d. From (1.1), it can be observed that the knowledge of the path loss exponent is crucial to correctly estimate the distance d, given the average received power. Moreover, wireless channels are affected by shadowing effects which cause signal power variations. Therefore, channel knowledge is a strong requirement to accurately estimate the distance, but its estimate in realistic (indoor) environments is a great challenge as well.

Time-based localization techniques rely on measurements of the ToF of signals travelling between nodes. If two synchronized nodes communicate, the node receiving the signal can determine the Time of Arrival (ToA) of the incoming signal from the timestamp of the sending node. If the nodes are not synchronized, Time Difference of Arrival (TDoA) techniques can be used. They are based on the estimation of the difference between the arrival times of signals traveling between pairs of nodes. The use of a time-based localization algorithm allows taking advantage of the features of UWB signals. As a matter of fact, it is known that, when using a time domain-based approach, the accuracy of the position estimate can be improved by increasing the effective bandwidth of the signal [50].

Given the range-difference measurements, various localization techniques have been proposed in the literature. Iterative techniques—such as Taylor series expansion [17], Gauss-Newton, steepest descent, Levenberg-Marquardt algorithm [36], graph-based methods [73], and methods based on metaheuristics [38]—require an accurate initial position estimate (which is often not available) and are computationally expensive. Therefore, various closed-form solutions have been proposed in the literature [60]. Among them, this dissertation mainly focuses on: two techniques based

on a Two-Stage Maximum-Likelihood (TSML) approach, namely the TSML-ToA [23] and the TSML-TDoA methods [8], and the Plane Intersection (PI) method [59]. These algorithms have been implemented to obtain simulation and experimental results presented later in this dissertation. As these methods have been the starting point from the existing literature, in the remainder of this section we summarize them.

1.3.1 Localization Scenario and Notation

Let us consider a three-dimensional environment—of course, similar considerations can be made in a two-dimensional space. We assume to know the positions of M nodes, denoted as Anchor Nodes (ANs), and we denote them as follows

$$\underline{s}_i = [x_i, y_i, z_i]^T \qquad i \in \{1, \ldots, M\}. \tag{1.2}$$

We then assume to have a given number of Target Nodes (TNs), either static or moving, with unknown positions. Our goal is to estimate their positions, given the known coordinates of the ANs. Let us denote as $\underline{u} = [x, y, z]^T$ the true (unknown) position of a generic TN and as $\hat{\underline{u}} = [\hat{x}, \hat{y}, \hat{z}]^T$ its estimated position. It is worth noting that, in this dissertation, the notation $[\hat{\star}]$ denotes the estimated value of $[\star]$.

The true and the estimated distances between the $i-$th AN and the considered TN can be expressed as

$$\begin{aligned} r_i &\triangleq ||\underline{u} - \underline{s}_i|| = \sqrt{(\underline{u} - \underline{s}_i)^T (\underline{u} - \underline{s}_i)} \qquad i \in \{1, \ldots, M\} \\ \hat{r}_i &\triangleq ||\hat{\underline{u}} - \underline{s}_i|| = \sqrt{(\hat{\underline{u}} - \underline{s}_i)^T (\hat{\underline{u}} - \underline{s}_i)} \qquad i \in \{1, \ldots, M\}. \end{aligned} \tag{1.3}$$

In the following, we assume that the errors which affect the range measurements $\{\hat{r}_i\}_{i=1}^M$ can be modeled as independent additive random variables $\{\nu_i\}_{i=1}^M$, namely

$$\hat{r}_i = r_i + \nu_i \qquad i \in \{1, \ldots, M\}. \tag{1.4}$$

Let us define as $\underline{\nu}$ the vector whose $i-$th element is ν_i and let us denote as $\underline{\underline{Q}}$ its covariance matrix.

We also denote as a_i the Euclidean norm of the coordinate vector of the $i-$th AN, namely

$$a_i \triangleq ||\underline{s}_i|| = \sqrt{x_i^2 + y_i^2 + z_i^2} \qquad i \in \{1, \ldots, M\}. \tag{1.5}$$

The estimated distances between the ANs and the TN lead to the following system of equations

$$\begin{cases} (\hat{x}-x_1)^2+(\hat{y}-y_1)^2+(\hat{z}-z_1)^2=\hat{r}_1^2 \\ \dots \\ (\hat{x}-x_M)^2+(\hat{y}-y_M)^2+(\hat{z}-z_M)^2=\hat{r}_M^2. \end{cases} \tag{1.6}$$

The system (1.6) is a quadratic system of M equations in the three unknowns $\hat{x}$, $\hat{y}$, and $\hat{z}$. Among the variety of approaches which can be used to solve system (1.6) in the following we describe three algorithms which are well known from the literature, namely: *(i)* the TSML-ToA method [60]; *(ii)* the TSML-TDoA method [8]; and *(iii)* the PI method [59].

1.3.2 Two-Stage Maximum-Likelihood Time-of-Arrival

In order to solve system (1.6), a two-step approach based on Maximum-Likelihood (ML) technique can be considered. This method has been proposed in [60] and we refer to it as TSML-ToA. The first stage relies, preliminarly, on introducing the new variable

$$\hat{n} \triangleq ||\underline{\hat{u}}||^2=\hat{x}^2+\hat{y}^2+\hat{z}^2 \tag{1.7}$$

so that system (1.6) can be rewritten, in matrix notation, as

$$\underline{\underline{G}}_1\,\underline{\hat{\omega}}_1=\underline{\hat{h}}_1 \tag{1.8}$$

where

$$\begin{aligned} \underline{\underline{G}}_1 &= -2\begin{pmatrix} x_1 & y_1 & z_1 & -0.5 \\ \vdots & \vdots & \vdots & \vdots \\ x_M & y_M & z_M & -0.5 \end{pmatrix} \\ \underline{\hat{\omega}}_1 &= \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{n} \end{pmatrix} \qquad \underline{\hat{h}}_1=\begin{pmatrix} \hat{r}_1^2-a_1^2 \\ \vdots \\ \hat{r}_M^2-a_M^2 \end{pmatrix}. \end{aligned} \tag{1.9}$$

While (1.8) might look like a linear system, it is not, since the fourth element of the solution vector $\hat{\underline{\omega}}_1$ depends on the first three according to (1.7). The solution $\hat{\underline{\omega}}_1$ of the system (1.8) can be determined through an ML approach. In particular, as suggested in [60], let us define the error vector

$$\underline{\psi}_1 \triangleq \hat{\underline{h}}_1 - \underline{\underline{G}}_1 \underline{\omega}_1. \tag{1.10}$$

Given a positive definite matrix $\underline{\underline{W}}_1$, the weighted Least Square (LS) solution of (1.8) that minimizes $\underline{\psi}_1^T \underline{\underline{W}}_1 \underline{\psi}_1$ is

$$\hat{\underline{\omega}}_1 = (\underline{\underline{G}}_1^T \underline{\underline{W}}_1 \underline{\underline{G}}_1)^{-1} \underline{\underline{G}}_1^T \underline{\underline{W}}_1 \hat{\underline{h}}_1. \tag{1.11}$$

The simplest choice of the weighting matrix $\underline{\underline{W}}_1$ is the identity matrix. In [60] it is shown that the choice of $\underline{\underline{W}}_1$ which minimizes the variance of $\hat{\underline{\omega}}_1$ is

$$\underline{\underline{W}}_1 \triangleq \mathrm{Cov}[\underline{\psi}_1]^{-1} = (4\underline{\underline{B}}\,\underline{\underline{Q}}\,\underline{\underline{B}})^{-1} \tag{1.12}$$

where $\underline{\underline{Q}}$ is the covariance matrix of the ToA range measurements $\{r_i\}_{i=1}^M$, $\underline{\underline{B}}$ is a diagonal matrix whose elements are $\{r_i\}_{i=1}^M$, and the last equality follows from the fact that, from (1.4) and (1.9), $\underline{\psi}_1$ can be written as

$$\underline{\psi}_1 = \hat{\underline{h}}_1 - \underline{h}_1 = 2\underline{\underline{B}}\,\underline{\nu} + \underline{\nu} \odot \underline{\nu} \simeq 2\underline{\underline{B}}\,\underline{\nu} \tag{1.13}$$

where $\odot$ denotes the entrywise product and the last approximation is obtained neglecting second order perturbations. With this choice of the weighting matrix one obtains that $\mathrm{Cov}[\hat{\underline{\omega}}_1] = (\underline{\underline{G}}_1^T \underline{\underline{W}}_1 \underline{\underline{G}}_1)^{-1}$.

The second stage of the algorithm is meant to take into account the dependence of $\hat{n}$ on the other unknowns of (1.8) and it involves the solution of the system

$$\underline{\underline{G}}_2 \hat{\underline{\omega}}_2 = \hat{\underline{h}}_2 \tag{1.14}$$

where

$$\underline{\underline{G}}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \hat{\underline{\omega}}_2 = \begin{pmatrix} \hat{x}^2 \\ \hat{y}^2 \\ \hat{z}^2 \end{pmatrix} \qquad \hat{\underline{h}}_2 = \begin{pmatrix} [\hat{\underline{\omega}}_1]_1^2 \\ [\hat{\underline{\omega}}_1]_2^2 \\ [\hat{\underline{\omega}}_1]_3^2 \\ [\hat{\underline{\omega}}_1]_4 \end{pmatrix} \tag{1.15}$$

and $[\hat{\underline{\omega}}_1]_j$ denotes the $j-$th component of $\hat{\underline{\omega}}_1$. The linear system (1.14) can be solved, once again, through the ML technique. Defining the error vector

$$\underline{\psi}_2 \triangleq \hat{\underline{h}}_2 - \underline{\underline{G}}_2 \, \underline{\omega}_2 \tag{1.16}$$

the weighted LS solution of (1.14) that minimizes the weighted norm of $\underline{\psi}_2$ with a positive definite matrix $\underline{\underline{W}}_2$ is

$$\hat{\underline{\omega}}_2 = (\underline{\underline{G}}_2^T \, \underline{\underline{W}}_2 \, \underline{\underline{G}}_2)^{-1} \underline{\underline{G}}_2^T \, \underline{\underline{W}}_2 \, \hat{\underline{h}}_2. \tag{1.17}$$

As considered to solve (1.8), the simplest choice of the weighting matrix $\underline{\underline{W}}_2$ is the identity matrix. In [60], it is shown that the choice of $\underline{\underline{W}}_2$ which minimizes the variance of $\hat{\underline{\omega}}_2$ is

$$\underline{\underline{W}}_2 \triangleq \mathrm{Cov}[\underline{\psi}_2]^{-1} = (4\underline{\underline{B}}_2 \, \mathrm{Cov}[\hat{\underline{\omega}}_1] \underline{\underline{B}}_2)^{-1} \tag{1.18}$$

where $\underline{\underline{B}}_2 = \mathrm{diag}(x, y, z, 0.5)$. Finally, the position estimate can be expressed as

$$\hat{\underline{u}} = \underline{\underline{U}} \left[\sqrt{[\hat{\underline{\omega}}_2]_1}, \sqrt{[\hat{\underline{\omega}}_2]_2}, \sqrt{[\hat{\underline{\omega}}_2]_3} \right]^T \tag{1.19}$$

where $\underline{\underline{U}} = \mathrm{diag}(\mathrm{sign}(\hat{\underline{\omega}}_1))$ [60].

1.3.3 Two-Stage Maximum-Likelihood Time Difference of Arrival

We preliminarily remark that, using a TDoA approach, it is known that the TSML method can reach the Cramer-Rao Lower Bound (CRLB), which is a generic lower bound for the variance of an estimator [24]. We refer to this algorithm as TSML-TDoA. In order to keep the notation tractable, let us define

$$x_{1j} \triangleq x_j - x_1 \qquad y_{1j} \triangleq y_j - y_1 \qquad z_{1j} \triangleq z_j - z_1 \qquad \forall j \in \{2, \ldots, M\}. \tag{1.20}$$

Similarly, let us denote the differences between the $j-$th and first range measurements, and their estimates, as

$$\Delta_{1j} \triangleq r_j - r_1 \qquad \hat{\Delta}_{1j} \triangleq \hat{r}_j - \hat{r}_1 \qquad \forall j \in \{2, \ldots, M\}. \tag{1.21}$$

From (1.4) one can then write

$$\hat{\Delta}_{1j} = \Delta_{1j} + \nu_j - \nu_1 \qquad \forall j \in \{2, \ldots, M\}. \tag{1.22}$$

Let us define

$$\underline{\nu}_1 \triangleq [\nu_2 - \nu_1, \ldots, \nu_M - \nu_1]^T \tag{1.23}$$

and let us denote as $\underline{\underline{Q}}$ its covariance matrix. Taking the difference between the $j-$th and the first equations of (1.6), one obtains a system of $M-1$ equations in the three unknowns $\hat{x}$, $\hat{y}$, and $\hat{z}$. The derivation of the solution begins by defining an auxiliary vector

$$\underline{\hat{\phi}}_1 \triangleq [\hat{x}, \hat{y}, \hat{z}, \hat{r}_1]^T. \tag{1.24}$$

From the equation system (1.6), after algebraic manipulations, the following system can be derived [24]

$$\underline{\underline{\hat{G}}}_1 \underline{\hat{\phi}}_1 = \underline{\hat{h}}_1 \tag{1.25}$$

where

$$\begin{aligned} \underline{\underline{\hat{G}}}_1 &= -2 \begin{pmatrix} x_{12} & y_{12} & z_{12} & \hat{\Delta}_{12} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1M} & y_{1M} & z_{1M} & \hat{\Delta}_{1M} \end{pmatrix} \\ \underline{\hat{\phi}}_1 &= \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{r}_1 \end{pmatrix} \qquad \underline{\hat{h}}_1 = \begin{pmatrix} a_1^2 - a_2^2 + \hat{\Delta}_{12}^2 \\ \vdots \\ a_1^2 - a_M^2 + \hat{\Delta}_{1M}^2 \end{pmatrix}. \end{aligned} \tag{1.26}$$

Observe that the system of equations (1.25) is non-linear because, according to (1.3), the fourth element of the solution vector depends on the first three ones. As in Subsection 1.3.2, the system (1.25) can be solved in two steps. Ignoring the dependence of $\hat{r}_1$ on $\hat{x}$, $\hat{y}$, and $\hat{z}$, the system (1.25) can be solved as if it were a linear one, by using the ML technique. In particular, defining the error vector

$$\underline{\psi}_1 \triangleq \underline{\hat{h}}_1 - \underline{\underline{\hat{G}}}_1 \underline{\phi}_1 \tag{1.27}$$

the weighted LS solution of (1.25) that minimizes the weighted norm of $\underline{\psi}_1$ is

$$\underline{\hat{\phi}}_1 = (\underline{\underline{\hat{G}}}_1^T \underline{\underline{W}}_1 \underline{\underline{\hat{G}}}_1)^{-1} \underline{\underline{\hat{G}}}_1^T \underline{\underline{W}}_1 \underline{\hat{h}}_1 \tag{1.28}$$

where $\underline{\underline{W}}_1$ is a positive definite matrix. The simplest choice of the weighting matrix $\underline{\underline{W}}_1$ is the identity matrix. In [24], it is shown that the choice of $\underline{\underline{W}}_1$ which minimizes the variance of $\hat{\underline{\phi}}_1$ is

$$\underline{\underline{W}}_1 \triangleq \mathrm{Cov}[\underline{\psi}_1]^{-1} \tag{1.29}$$

Observe that, from (1.27), using (1.22), $\underline{\psi}_1$ can be written as

$$\underline{\psi}_1 = \hat{\underline{h}}_1 - \underline{h}_1 + 2r_1\underline{\nu}_1 \tag{1.30}$$

and inserting (1.26) into (1.30) one obtains

$$\underline{\psi}_1 = 2\underline{\underline{B}}\,\underline{\nu}_1 + \underline{\nu}_1 \odot \underline{\nu}_1 \simeq 2\underline{\underline{B}}\,\underline{\nu}_1 \tag{1.31}$$

where $\underline{\underline{B}}$ is a $(M-1)\times(M-1)$ diagonal matrix whose elements are $\{r_i\}_{i=2}^{M}$ and the last approximation is obtained neglecting second order perturbations. Hence, it can be concluded that

$$\underline{\underline{W}}_1 = (4\underline{\underline{B}}\,\underline{\underline{Q}}\,\underline{\underline{B}})^{-1} \tag{1.32}$$

where $\underline{\underline{Q}}$ is the covariance matrix of the TDoA distance measurements. With this choice of the weighting matrix one obtains that $\mathrm{Cov}[\hat{\underline{\phi}}_1] = (\hat{\underline{\underline{G}}}_1^T\,\underline{\underline{W}}_1\,\hat{\underline{\underline{G}}}_1)^{-1}$.

Given the solution $\hat{\underline{\phi}}_1$, the second step is meant to take into account that the elements of the solution vector are not independent, rather they are related to each other according to

$$[\hat{\underline{\phi}}_1]_4^2 = ([\hat{\underline{\phi}}_1]_1 - x_1)^2 + ([\hat{\underline{\phi}}_1]_2 - y_1)^2 + ([\hat{\underline{\phi}}_1]_3 - z_1)^2. \tag{1.33}$$

In order to take this dependence into account and to improve the TN position estimate accuracy, the following system of equations can be derived

$$\underline{\underline{G}}_2\,\hat{\underline{\phi}}_2 = \hat{\underline{h}}_2 \tag{1.34}$$

where $\underline{\underline{G}}_2$ is the same as in (1.15) and

$$\hat{\underline{\phi}}_2 = \begin{pmatrix} (\hat{x}-x_1)^2 \\ (\hat{y}-y_1)^2 \\ (\hat{z}-z_1)^2 \end{pmatrix} \qquad \hat{\underline{h}}_2 = \begin{pmatrix} ([\hat{\underline{\phi}}_1]_1 - x_1)^2 \\ ([\hat{\underline{\phi}}_1]_2 - y_1)^2 \\ ([\hat{\underline{\phi}}_1]_3 - z_1)^2 \\ [\hat{\underline{\phi}}_1]_4^2 \end{pmatrix}. \tag{1.35}$$

Defining the error vector

$$\underline{\psi}_2 \triangleq \hat{\underline{h}}_2 - \underline{\underline{G}}_2\,\underline{\phi}_2 \tag{1.36}$$

the weighted LS solution of (1.34) that minimizes the weighted norm of $\underline{\psi}_2$ with a positive definite matrix $\underline{\underline{W}}_2$ is

$$\hat{\underline{\phi}}_2 = (\underline{\underline{G}}_2^T\,\underline{\underline{W}}_2^{-1}\,\underline{\underline{G}}_2)^{-1}\underline{\underline{G}}_2^T\,\underline{\underline{W}}_2^{-1}\,\hat{\underline{h}}_2. \tag{1.37}$$

The choice of $\underline{\underline{W}}_2$ which minimizes the variance of $\hat{\underline{\omega}}_2$ is

$$\underline{\underline{W}}_2 \triangleq \mathrm{Cov}[\underline{\psi}_2]^{-1} = (4\underline{\underline{B}}_2\,\mathrm{Cov}[\hat{\underline{\phi}}_1]\,\underline{\underline{B}}_2)^{-1} \tag{1.38}$$

where $\underline{\underline{B}}_2 = \mathrm{diag}(x-x_1, y-y_1, z-z_1, r_1)$ [23]. Finally, the solution $\hat{\underline{\phi}}_2$ is used to obtain the following position estimate of the target

$$\hat{\underline{u}} = \underline{\underline{U}}\left[\sqrt{[\hat{\underline{\phi}}_2]_1}, \sqrt{[\hat{\underline{\phi}}_2]_2}, \sqrt{[\hat{\underline{\phi}}_2]_3}\right]^T + \underline{s}_1$$

where $\underline{\underline{U}} = \mathrm{diag}[\mathrm{sgn}(\hat{\underline{\phi}}_1 - \underline{s}_1)]$.

1.3.4 Plane Intersection

With the PI method [59], each triple of ANs yields a plane on which the TN lies and the position estimate is obtained by intersecting such planes, with equations depending on the reference ANs. More precisely, three triples of ANs yield three planes, the intersection of which is a single point which coincides with the TN position. Then, if $M \geq 5$, considering, for instance, the $M-2$ triples of ANs given by $\{(\underline{s}_1, \underline{s}_2, \underline{s}_j)\}_{j=3}^{M}$, using the notation defined in (1.20) and (1.21), the $M-2$ planes are given by [59]

$$\begin{aligned}\mathscr{P}^{(1,2,j)} = \Bigg\{ & (x,y,z) \in \mathbb{R}^3 : (x_{1j}\hat{\Delta}_{12} - x_{12}\hat{\Delta}_{1j})x + (y_{1j}\hat{\Delta}_{12} - y_{12}\hat{\Delta}_{1j})y \\ & + (z_{1j}\hat{\Delta}_{12} - z_{12}\hat{\Delta}_{1j})z \\ & = -\frac{1}{2}\hat{\Delta}_{12}\hat{\Delta}_{1j}(\hat{\Delta}_{1j} - \hat{\Delta}_{12}) + \frac{1}{2}(a_1^2 - a_2^2)\hat{\Delta}_{1j} \\ & - \frac{1}{2}(a_1^2 - a_j^2)\hat{\Delta}_{12} \Bigg\} \qquad j \in \{3,\ldots,M\}.\end{aligned} \tag{1.39}$$

It can thus be concluded that the solution vector $\hat{\underline{u}} \triangleq [\hat{x},\hat{y},\hat{z}]^T$ of the estimated coordinates of the TN has to satisfy the following system of equations

$$\hat{\underline{\underline{A}}}\hat{\underline{u}} = \hat{\underline{b}} \tag{1.40}$$

where

$$\hat{\underline{\underline{A}}} = \begin{pmatrix} x_{13}\hat{\Delta}_{12} - x_{12}\hat{\Delta}_{13} & y_{13}\hat{\Delta}_{12} - y_{12}\hat{\Delta}_{13} & z_{13}\hat{\Delta}_{12} - z_{12}\hat{\Delta}_{13} \\ \vdots & \vdots & \vdots \\ x_{12}\hat{\Delta}_{1M} - x_{1M}\hat{\Delta}_{12} & y_{12}\hat{\Delta}_{1M} - y_{1M}\hat{\Delta}_{12} & z_{12}\hat{\Delta}_{1M} - z_{1M}\hat{\Delta}_{12} \end{pmatrix} \tag{1.41}$$

and

$$\hat{\underline{b}} = \frac{1}{2}\begin{pmatrix} \hat{\Delta}_{12}\hat{\Delta}_{13}(\hat{\Delta}_{12} - \hat{\Delta}_{13}) + (a_1^2 - a_2^2)\hat{\Delta}_{13} - (a_1^2 - a_3^2)\hat{\Delta}_{12} \\ \vdots \\ \hat{\Delta}_{12}\hat{\Delta}_{1M}(\hat{\Delta}_{12} - \hat{\Delta}_{1M}) + (a_1^2 - a_2^2)\hat{\Delta}_{1M} - (a_1^2 - a_M^2)\hat{\Delta}_{12} \end{pmatrix}. \tag{1.42}$$

The LS solution of the system of equations (1.40) is then

$$\hat{\underline{u}} = (\underline{\underline{A}}^T\underline{\underline{A}})^{-1}\underline{\underline{A}}^T\underline{b}. \tag{1.43}$$

In Chapter 2, a variant of the PI method will be described for a more specific localization problem.

Chapter 2

An Analytical Approach to Optimized Anchors Placement

Longum iter est per praecepta,
breve et efficax per exempla

– Lucius Annaeus Seneca

2.1 Introduction

In this chapter, we consider the problem of localizing a TN moving along a corridor in a (large) indoor environment by means of UWB signaling from fixed ANs and we focus on optimal ANs positioning. The problem of optimal sensor placement for effective distributed processing has been studied in the literature in various contexts [3, 33]. The aim of this chapter is to derive an analytical approach to optimized ANs placement, according to the criterion of minimizing, under proper realistic constraints, the Root Mean Square Error (RMSE) between the position estimate and the true position of the TN.

We consider a three-dimensional scenario where the TN moves on the floor along a corridor and the ANs are positioned uniformly on both sides of the corridor at the same height [44, 48]. Such a scenario can be found, for instance, in industrial environment where shelves create corridors in which people and vehicles move and where the ANs can be placed on the ceiling, at the top of shelves.

The PI algorithm is chosen because it allows deriving a closed-form expression for the inter-AN distance in the considered scenario. Using four ANs it is possible to estimate the TN position by intersecting the major axes of two three-dimensional conics associated with two subsets of three ANs.

Under the assumption of a fixed variance of the range estimation error, we derive here a simple closed-form expression for the optimal inter-AN distance in terms of the corridor width and the height of the ANs. Simulation results confirm the effectiveness of the proposed analytical approach to optimized ANs placement. It is shown that the proposed approach allows the MSE of the TN position estimates to reach the CRLB.

2.2 Localization Using Four ANs: General Framework

In the remainder of this chapter, it is assumed that the TN moves on the floor e.g., of a warehouse. Without loss of generality, let this plane be given by

$$\mathscr{P}_0 \triangleq \{(x,y,z) \in \mathbb{R}^3 : z = 0\}. \tag{2.1}$$

According to [59], the minimum number of ANs needed by the PI method to estimate the TN position in a three-dimensional scenario is five. However, in the considered scenario, one of the coordinates of the TN is known ($z = 0$) so this number reduces to four. Suppose that the position estimate of the TN is obtained by using the four closest ANs which are in LoS with the TN. Using the same notation introduced in (1.2), the ANs coordinates are denoted as

$$\underline{s}_i = [x_i, y_i, z_i]^T \qquad i \in \{1,2,3,4\}.$$

As soon as the TN receives signals from the four ANs, it can localize itself by either processing the received signals on board or by sending the data to a server which

estimates the position. Let $\underline{u} = [x, y, 0]^T$ and $\hat{\underline{u}} = [\hat{x}, \hat{y}, 0]^T$ be the true and estimated coordinates of the TN, respectively. Then, the true distances $\{r_i\}_{i=1}^4$ and the estimated distances $\{\hat{r}_i\}_{i=1}^4$ between the TN and the considered ANs can be written as in (1.3).

According to [24], the ToA measurements can be described by an additive noise model, so that the estimated distances $\{\hat{r}_i\}_{i=1}^4$ can be expressed as

$$\hat{r}_i = r_i + v_i \qquad i \in \{1,2,3,4\} \tag{2.2}$$

where $\{v_i\}_{i=1}^4$ are the distance errors. In [5], it is shown that, with UWB signaling, the range error can be written as follows

$$v_i = \varepsilon_i + \beta \qquad i \in \{1,2,3,4\} \tag{2.3}$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$; ε_i is independent of ε_j for $i \neq j$; and β is the synchronization bias (the same for all the ANs). One can thus write

$$\underline{\varepsilon} \sim \mathcal{N}(\underline{0}, \mathrm{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2)) \tag{2.4}$$

where $\underline{\varepsilon} \triangleq [\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4]^T$.

As in (1.21), let us denote as $\{\Delta_{1j}\}_{j=2}^4$ and $\{\hat{\Delta}_{1j}\}_{j=2}^4$ the true and the estimated range-differences between the first and the $j-$th AN, respectively. All other possibile range-differences can then be expressed in terms of $\{\Delta_{1j}\}_{j=2}^4$ and $\{\hat{\Delta}_{1j}\}_{j=2}^4$. Inserting (2.2) into the definitions of $\{\Delta_{1j}\}_{j=2}^4$ and $\{\hat{\Delta}_{1j}\}_{j=2}^4$ leads to

$$\hat{\Delta}_{1j} = \Delta_{1j} + \varepsilon_{1j} \qquad j \in \{2,3,4\} \tag{2.5}$$

where $\varepsilon_{1j} \triangleq \varepsilon_j - \varepsilon_1$ is the error in the estimated range-difference between the first and the $j-$th AN. From (2.4), it follows that

$$\begin{aligned} &\mathbb{E}[\varepsilon_{1j}] = \mathbb{E}[\varepsilon_j] - \mathbb{E}[\varepsilon_1] = 0 && j \in \{2,3,4\} \\ &\mathbb{E}[\varepsilon_{1j}\varepsilon_{1k}] = \mathbb{E}[\varepsilon_j\varepsilon_k] + \mathbb{E}[\varepsilon_1^2] = \sigma_j^2\delta_{jk} + \sigma_1^2 && j,k \in \{2,3,4\} \end{aligned}$$

where δ_{jk} is the Kronecker symbol and $\mathbb{E}$ is the expectation operator. By defining

$$\underline{\varepsilon}_1 \triangleq [\varepsilon_{12}, \varepsilon_{13}, \varepsilon_{14}]^T \tag{2.6}$$

it can be concluded that $\underline{\varepsilon}_1 \sim \mathscr{N}(\underline{0}, \underline{\underline{Q}})$, where

$$\underline{\underline{Q}} = \begin{pmatrix} \sigma_1^2+\sigma_2^2 & \sigma_1^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2+\sigma_3^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 & \sigma_1^2+\sigma_4^2 \end{pmatrix}. \tag{2.7}$$

With the PI method [59], the position estimate is obtained by intersecting planes with equations depending on the reference ANs. More precisely, two triples of ANs yield two planes, the intersection of which is a line where the TN lies.

Once the equation of such a line is known, it is sufficient to intersect it with the plane $\mathscr{P}_0$ defined in (2.1) in order to obtain the estimated position of the TN. Considering, for instance, the two triples of ANs given by $\{\underline{s}_1, \underline{s}_2, \underline{s}_3\}$ and $\{\underline{s}_1, \underline{s}_2, \underline{s}_4\}$, and assuming the notation defined in (1.5) and (1.20), the two planes are given by [59]

$$\begin{aligned} \mathscr{P}^{(1,2,j)} = \Bigg\{ (x,y,z) \in \mathbb{R}^3 : {} & (x_{1j}\hat{\Delta}_{12} - x_{12}\hat{\Delta}_{1j})x + (y_{1j}\hat{\Delta}_{12} - y_{12}\hat{\Delta}_{1j})y \\ & + (z_{1j}\hat{\Delta}_{12} - z_{12}\hat{\Delta}_{1j})z \\ & = -\frac{1}{2}\hat{\Delta}_{12}\hat{\Delta}_{1j}(\hat{\Delta}_{1j} - \hat{\Delta}_{12}) + \frac{1}{2}(a_1^2 - a_2^2)\hat{\Delta}_{1j} \\ & - \frac{1}{2}(a_1^2 - a_j^2)\hat{\Delta}_{12} \Bigg\} \qquad j \in \{3,4\}. \end{aligned}$$

Since it is assumed that the TN moves on the plane $\mathscr{P}_0$ defined in (2.1), it can thus be concluded that the vector $\hat{\underline{u}}_{[1,2]} \triangleq [\hat{x}, \hat{y}]^T$ of the estimated x and y coordinates of the TN has to satisfy the following system of equations

$$\hat{\underline{\underline{A}}}\,\hat{\underline{u}}_{[1,2]} = \hat{\underline{b}} \tag{2.8}$$

where

$$\hat{\underline{\underline{A}}} = \begin{pmatrix} x_{13}\hat{\Delta}_{12} - x_{12}\hat{\Delta}_{13} & y_{13}\hat{\Delta}_{12} - y_{12}\hat{\Delta}_{13} \\ x_{14}\hat{\Delta}_{12} - x_{12}\hat{\Delta}_{14} & y_{14}\hat{\Delta}_{12} - y_{12}\hat{\Delta}_{14} \end{pmatrix} \tag{2.9}$$

and

$$\hat{\underline{b}} = \frac{1}{2}\begin{pmatrix} \hat{\Delta}_{12}\hat{\Delta}_{13}(\hat{\Delta}_{12} - \hat{\Delta}_{13}) + (a_1^2 - a_2^2)\hat{\Delta}_{13} - (a_1^2 - a_3^2)\hat{\Delta}_{12} \\ \hat{\Delta}_{12}\hat{\Delta}_{14}(\hat{\Delta}_{12} - \hat{\Delta}_{14}) + (a_1^2 - a_2^2)\hat{\Delta}_{14} - (a_1^2 - a_4^2)\hat{\Delta}_{12} \end{pmatrix}. \tag{2.10}$$

Similarly, the actual coordinates $\underline{u}_{[1,2]} \triangleq [x,y]^T$ of the TN on $\mathscr{P}_0$ satisfy the following system of equations

$$\underline{\underline{A}}\,\underline{u}_{[1,2]} = \underline{b} \tag{2.11}$$

where $\underline{\underline{A}}$ and $\underline{b}$ are obtained by substituting the estimated range-differences $\{\hat{\Delta}_{1j}\}_{j=2}^{4}$ with the true ones $\{\Delta_{1j}\}_{j=2}^{4}$ in $\hat{\underline{\underline{A}}}$ and $\hat{\underline{b}}$.

From (2.5), one can write

$$\hat{\underline{\underline{A}}} = \underline{\underline{A}} + \underline{\underline{E}} \qquad \hat{\underline{b}} = \underline{b} + \underline{e}_{\underline{b}} \tag{2.12}$$

where

$$\underline{\underline{E}} \triangleq \begin{pmatrix} x_{13}\varepsilon_{12} - x_{12}\varepsilon_{13} & y_{13}\varepsilon_{12} - y_{12}\varepsilon_{13} \\ x_{14}\varepsilon_{12} - x_{12}\varepsilon_{14} & y_{14}\varepsilon_{12} - y_{12}\varepsilon_{14} \end{pmatrix} \tag{2.13}$$

and

$$\begin{aligned} [\underline{e}_{\underline{b}}]_i \triangleq\ & \frac{1}{2}\varepsilon_{1i+2}[{\Delta_{12}}^2 - 2\Delta_{12}\Delta_{1i+2} + a_1^2 - a_2^2] \\ & - \frac{1}{2}\varepsilon_{12}[{\Delta_{1i+2}}^2 - 2\Delta_{12}\Delta_{1i+2} + a_1^2 - a_{i+2}^2] \\ & + \frac{1}{2}\varepsilon_{12}\varepsilon_{1i+2}[2\Delta_{12} - 2\Delta_{1i+2}] + \varepsilon_{12}^2\Delta_{1i+2} \\ & + \varepsilon_{1i+2}^2\Delta_{12} + \varepsilon_{12}\varepsilon_{1i+2}(\varepsilon_{12} - \varepsilon_{1i+2}) \qquad i \in \{1,2\}. \end{aligned} \tag{2.14}$$

Assuming that the components of vector $\underline{\varepsilon}_1$ defined in (2.6) are small (which is realistic with UWB signaling), a good approximation of $\underline{e}_{\underline{b}}$ can be obtained by omitting non-linear perturbations in (2.14). Therefore, it can be stated that

$$\hat{\underline{b}} \simeq \underline{b} + \underline{\varepsilon}_{\underline{b}} \tag{2.15}$$

where the $i-$th component of $\underline{\varepsilon}_{\underline{b}}$ is

$$\begin{aligned} [\underline{\varepsilon}_{\underline{b}}]_i = \frac{1}{2}\Big(& \varepsilon_{1i+2}[\Delta_{12}^2 - 2\Delta_{12}\Delta_{1i+2} + a_1^2 - a_2^2] \\ & - \varepsilon_{12}[\Delta_{1i+2}^2 - 2\Delta_{12}\Delta_{1i+2} + a_1^2 - a_{i+2}^2]\Big). \end{aligned} \tag{2.16}$$

Define the auxiliary vector

$$\underline{\psi} \triangleq \hat{\underline{b}} - \hat{\underline{\underline{A}}}\,\underline{u}_{[1,2]} = \hat{\underline{\underline{A}}}\,\underline{e} \tag{2.17}$$

where $\underline{e} \triangleq \underline{\hat{u}}_{[1,2]} - \underline{u}_{[1,2]}$ is the difference between the true position of the TN on $\mathscr{P}_0$ and its estimated position, i.e., $\underline{e}$ is the position estimation error. Using (2.12) and (2.15), (2.17) can be written as

$$\underline{\psi} \simeq \underline{b} + \underline{\varepsilon}_{\underline{b}} - (\underline{\underline{A}} + \underline{\underline{E}})\,\underline{u}_{[1,2]} = \underline{\varepsilon}_{\underline{b}} - \underline{\underline{E}}\,\underline{u}_{[1,2]} \tag{2.18}$$

where the last equality follows from (2.11). Substituting (2.13) and (2.16) into (2.18), the $i-$th component of the vector $\underline{\psi}$ can be approximated as

$$\begin{aligned}
\psi_i &\simeq \frac{1}{2}\left[\varepsilon_{1i+2}\left(\Delta_{12}^2 - 2\Delta_{12}\Delta_{1i+2}\right) - \varepsilon_{12}\left(\Delta_{1i+2}^2 - 2\Delta_{12}\Delta_{1i+2}\right)\right] \\
&+ \frac{1}{2}\varepsilon_{1i+2}\left(a_1^2 - a_2^2 - 2x_{21}x - 2y_{21}y\right) - \frac{1}{2}\varepsilon_{12}\left(a_1^2 - a_{i+2}^2 - 2x_{i+21}x - 2y_{i+21}y\right) \\
&= \frac{1}{2}\left[\varepsilon_{1i+2}\left(\Delta_{12}^2 - 2\Delta_{12}\Delta_{1i+2}\right) - \varepsilon_{12}\left(\Delta_{1i+2}^2 - 2\Delta_{12}\Delta_{1i+2}\right)\right] \\
&+ \frac{1}{2}\varepsilon_{1i+2}\left(r_1^2 - r_2^2\right) - \frac{1}{2}\varepsilon_{12}\left(r_1^2 - r_{i+2}^2\right) \qquad i \in \{1,2\}.
\end{aligned}$$

After substituting (1.21) into the previous equation and some manipulations, one obtains the following approximate expressions for the elements of vector $\underline{\psi}$

$$\begin{aligned}
\psi_1 &\simeq \varepsilon_{13}\, r_3\,(r_1 - r_2) - \varepsilon_{12}\, r_2\,(r_1 - r_3) \\
\psi_2 &\simeq \varepsilon_{14}\, r_4\,(r_1 - r_2) - \varepsilon_{12}\, r_2\,(r_1 - r_4).
\end{aligned}$$

Using (1.21) and (2.6), the above equations can be written as $\underline{\psi} \simeq \underline{\underline{R}}\,\underline{\varepsilon}_1$, where

$$\underline{\underline{R}} \triangleq \begin{pmatrix} r_2\Delta_{13} & -r_3\Delta_{12} & 0 \\ r_2\Delta_{14} & 0 & -r_4\Delta_{12} \end{pmatrix}. \tag{2.19}$$

Since $\mathbb{E}[\underline{\varepsilon}_1] = \underline{0}$, it follows that $\mathbb{E}[\underline{\psi}] = \underline{0}$ and the covariance matrix of $\underline{\psi}$ is

$$\underline{\underline{\Psi}} \triangleq \mathbb{E}[\underline{\psi}\,\underline{\psi}^T] \simeq \underline{\underline{R}}\,\mathbb{E}[\underline{\varepsilon}_1\underline{\varepsilon}_1^T]\,\underline{\underline{R}}^T = \underline{\underline{R}}\,\underline{\underline{Q}}\,\underline{\underline{R}}^T$$

where $\underline{\underline{Q}}$ is defined in (2.7).

From (2.17), assuming that $\det\underline{\underline{\hat{A}}} \neq 0$, the error $\underline{e}$ can be written as

$$\underline{e} = \underline{\underline{\hat{A}}}^{-1}\,\underline{\psi} \simeq (\underline{\underline{A}} + \underline{\underline{E}})^{-1}\,\underline{\underline{R}}\,\underline{\varepsilon}_1.$$

Neglecting, once again, non-linear perturbations, the error $\underline{e}$ can be approximated as follows

$$\underline{e} \simeq \underline{\underline{A}}^{-1} \underline{\underline{R}} \underline{\varepsilon}_1. \tag{2.20}$$

Therefore, by defining

$$\begin{aligned} \underline{\underline{B}} &\triangleq \underline{\underline{A}}^{-1} \\ \underline{\underline{T}} &\triangleq \underline{\underline{B}}\underline{\underline{R}} \end{aligned} \tag{2.21}$$

one can finally conclude that $\underline{e} \sim \mathcal{N}(\underline{0}, \underline{\underline{C}})$, where

$$\underline{\underline{C}} = \mathrm{Cov}[\underline{e}] \simeq \mathbb{E}[\underline{\underline{T}}\,\underline{\varepsilon}_1\,\underline{\varepsilon}_1^T\,\underline{\underline{T}}^T] = \underline{\underline{T}}\,\underline{\underline{Q}}\,\underline{\underline{T}}^T. \tag{2.22}$$

The trace of $\underline{\underline{C}}$ is the Mean Square Error (MSE) of the position estimation and it can be expressed, as a function of $\underline{\underline{T}}$ and $\underline{\underline{Q}}$, as follows

$$Tr(\underline{\underline{C}}) \simeq \sigma_1^2 Tr(\underline{\underline{T}}\,\underline{\underline{1}}_3\,\underline{\underline{T}}^T) + Tr(\underline{\underline{T}}\,\mathrm{diag}([\sigma_2^2, \sigma_3^2, \sigma_4^2])\underline{\underline{T}}^T) \tag{2.23}$$

where $\underline{\underline{1}}_3$ is a 3×3 matrix with all elements equal to 1. Simple matrix calculation leads to the following approximation

$$\begin{aligned} Tr(\underline{\underline{C}}) \simeq\ & \sigma_1^2 \left[(T_{11} + T_{12} + T_{13})^2 + (T_{21} + T_{22} + T_{23})^2 \right] \\ & + \sigma_2^2 (T_{11}^2 + T_{21}^2) + \sigma_3^2 (T_{12}^2 + T_{22}^2) + \sigma_4^2 (T_{13}^2 + T_{23}^2). \end{aligned} \tag{2.24}$$

The analytical expression of $Tr(\underline{\underline{C}})$ obtained in (2.24) shows that $Tr(\underline{\underline{C}})$ depends on the coordinates of the ANs and on the distances between each AN and the TN.

In Section 2.3, $Tr(\underline{\underline{C}})$ is explicitly calculated, under realistic assumptions, in a specific scenario where the TN follows a straight path in the middle of a corridor, and optimized placement of ANs is performed by the analytical minimization of the $Tr(\underline{\underline{C}})$. Moreover, Section 2.4 shows that the proposed optimized placement strategy provides accurate and satisfactory position estimates even if the TN follows a straight path not in the middle of the corridor or even a non-straight path.

2.3 Position Estimation of a TN Moving on a Line

Suppose that the TN moves along a straight line in the middle of a corridor of width ω, as shown in Figure 2.1, so that its position at time t can be expressed as

$$\underline{u}(t) = \left[x(t), \frac{\omega}{2}, 0\right]^T.$$

We assume that the ANs are alternately positioned on the two sides of the corridor and that they are regularly spaced—this is realistic from an installation perspective, for large indoor scenarios. In particular, as shown in Figure 2.1, we denote as 2δ the distance between two consecutive ANs on the same side of the corridor along the $x-$axis (so that δ is the difference along the $x-$axis between two consecutive ANs on opposite sides of the corridor).

Finally, we also assume that all the ANs are placed at the same height $\zeta \geq 0$, i.e. on the same plane (such as the ceiling of the building) parallel to $\mathscr{P}_0$ (on which the TN moves), given by $\mathscr{P}_\zeta \triangleq \{(x,y,z) \in \mathbb{R}^3 : z = \zeta\}$. It can be shown that, without knowing that the TN moves on a specific plane, the assumption of all ANs located on the same plane would not lead to a unique solution for the TN location problem. The above assumptions, which may appear very stringent, are realistic in several industrial scenarios where shelving units identify straight corridors, along which TNs (e.g., automated guided vehicles) move.

As the TN moves along its trajectory, it dynamically selects the four closest ANs in LoS to estimate its position, and thus, the framework outlined in Section 2.2 can be applied. Except for an initial transitory (when $0 \leq x(t) \leq \delta$) and for the end of the corridor, the considered AN configuration along the corridor is periodic. We remark that the estimation strategy does not change at the beginning and at the end of the corridor, as the four nearest ANs are used. The only difference consists in a different configuration on the ANs with respect to the TN. Therefore, without loss of generality we restrict our attention to the interval $\delta \leq x(t) \leq 2\delta$, for which the coordinates of the four nearest ANs are given by

$$\underline{s}_1 = [0,0,\zeta]^T \qquad \underline{s}_2 = [\delta,\omega,\zeta]^T \qquad \underline{s}_3 = [2\delta,0,\zeta]^T \qquad \underline{s}_4 = [3\delta,\omega,\zeta]^T.$$

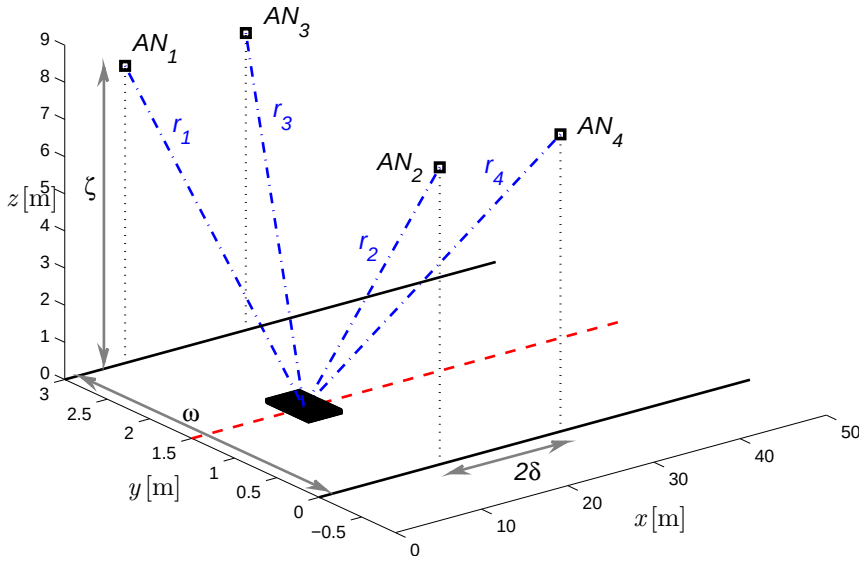


Figure 2.1: A corridor in a general scenario where the TN, represented by the black rectangle, moves. The solid black lines on the floor represent the projections of walls on $\mathscr{P}_0$, thus indicating the width of the corridor. The four nearest ANs (*black squares*) and their distances $\{r_i\}_{i=1}^4$ (*dash-dotted blue lines*) from the TN are also indicated.

If $\zeta = 0$, then all the ANs are located on the same plane where the TN lies, and this scenario reduces to the one considered in [40].

Given the width ω of the corridor and the height ζ of the ANs, our goal is to compute $Tr(\underline{\underline{C}})$ and to find the distance δ^* that minimizes it. To determine an expression for $\underline{\underline{C}}$, explicit expressions of the distances between the TN and the ANs are needed.

Denoting the position of the TN at a given instant as $[\bar{x}, \omega/2, 0]^T$, one can write

$$\begin{aligned} r_1^2 &= \bar{x}^2 + \left(\tfrac{\omega}{2}\right)^2 + \zeta^2 & r_2^2 &= (\delta - \bar{x})^2 + \left(\tfrac{\omega}{2}\right)^2 + \zeta^2 \\ r_3^2 &= (2\delta - \bar{x})^2 + \left(\tfrac{\omega}{2}\right)^2 + \zeta^2 & r_4^2 &= (3\delta - \bar{x})^2 + \left(\tfrac{\omega}{2}\right)^2 + \zeta^2. \end{aligned} \tag{2.25}$$

Defining

$$\eta^2 \triangleq 4\left[\left(\frac{\omega}{2}\right)^2 + \zeta^2\right] \tag{2.26}$$

and substituting (2.26) into (2.25) gives

$$\begin{aligned} r_1 &= \frac{\eta}{2}\sqrt{1+\left(\frac{2\bar{x}}{\eta}\right)^2} & r_2 &= \frac{\eta}{2}\sqrt{1+\left(\frac{2(\delta-\bar{x})}{\eta}\right)^2} \\ r_3 &= \frac{\eta}{2}\sqrt{1+\left(\frac{2(2\delta-\bar{x})}{\eta}\right)^2} & r_4 &= \frac{\eta}{2}\sqrt{1+\left(\frac{2(3\delta-\bar{x})}{\eta}\right)^2}. \end{aligned} \tag{2.27}$$

If $\eta > 4\delta$, then (since $\delta \leq \bar{x} \leq 2\delta$) the second-order Taylor series expansion

$$\sqrt{1+\chi^2} = 1 + \chi^2/2 + o(\chi^3)$$

can be effectively applied to (2.27), which gives

$$\begin{aligned} r_1 &\simeq \frac{\eta}{2} + \frac{\bar{x}^2}{\eta} & r_2 &\simeq \frac{\eta}{2} + \frac{(\delta-\bar{x})^2}{\eta} \\ r_3 &\simeq \frac{\eta}{2} + \frac{(2\delta-\bar{x})^2}{\eta} & r_4 &\simeq \frac{\eta}{2} + \frac{(3\delta-\bar{x})^2}{\eta}. \end{aligned} \tag{2.28}$$

Note that the approximate expressions (2.28) are more accurate for larger η (i.e., larger ω and/or ζ) which corresponds to large indoor scenarios, one of our basic assumptions. According to (2.26), the condition $\eta > 4\delta$ can be written as

$$\delta < \frac{1}{2}\sqrt{(\omega/2)^2+\zeta^2}. \tag{2.29}$$

Therefore, the upper bound for δ increases as the width ω of the corridor or the height ζ of the ANs increase.

Substituting (2.28) into (1.21) gives

$$\Delta_{12} \simeq \frac{\delta(\delta-2\bar{x})}{\eta} \qquad \Delta_{13} \simeq \frac{4\delta(\delta-\bar{x})}{\eta} \qquad \Delta_{14} \simeq \frac{\delta(9\delta-6\bar{x})}{\eta}. \tag{2.30}$$

Using (2.30) and (2.9), the following approximate expression of $\underline{\underline{A}}$ can be derived

$$\underline{\underline{A}} \simeq \begin{pmatrix} -2\dfrac{\delta^3}{\eta} & -4\dfrac{\delta(\delta-\bar{x})\omega}{\eta} \\ -6\dfrac{\delta^3}{\eta} & -4\dfrac{\delta(2\delta-\bar{x})\omega}{\eta} \end{pmatrix} \tag{2.31}$$

so that its determinant is

$$\det \underline{\underline{A}} \simeq -8\omega \frac{\delta^4}{\eta^2}(\delta - 2\bar{x}).$$

Since $\delta \leq \bar{x} \leq 2\delta$, it can be concluded that $\det \underline{\underline{A}} > 0$ and, therefore, the matrix $\underline{\underline{B}}$ and the error $\underline{e}$ are well defined.

Before calculating the covariance matrix $\underline{\underline{C}}$ of $\underline{e}$, an explicit expression of $\underline{\underline{T}}$ (and, therefore, of $\underline{\underline{B}}$ and $\underline{\underline{R}}$) is needed. From (2.31) it follows that

$$\underline{\underline{B}} \simeq \frac{1}{\delta - 2\bar{x}} \begin{pmatrix} \frac{\eta}{2\delta^3}(2\delta - \bar{x}) & -\frac{\eta}{2\delta^3}(\delta - \bar{x}) \\ -\frac{3\eta}{4\delta\omega} & \frac{\eta}{4\delta\omega} \end{pmatrix}. \tag{2.32}$$

From (2.19), using (2.28) and (2.30), the entries of $\underline{\underline{R}}$ can be approximated as

$$\begin{aligned} R_{11} &\simeq 4\delta(\delta - \bar{x}) \left[\frac{1}{2} + \frac{(\delta - \bar{x})^2}{\eta^2}\right] \\ R_{12} &\simeq -\delta(\delta - 2\bar{x}) \left[\frac{1}{2} + \frac{(2\delta - \bar{x})^2}{\eta^2}\right] \\ R_{21} &\simeq 3\delta(3\delta - 2\bar{x}) \left[\frac{1}{2} + \frac{(\delta - \bar{x})^2}{\eta^2}\right] \\ R_{23} &\simeq -\delta(\delta - 2\bar{x}) \left[\frac{1}{2} + \frac{(3\delta - \bar{x})^2}{\eta^2}\right]. \end{aligned} \tag{2.33}$$

The entries of matrix $\underline{\underline{T}}$ defined in (2.21) can finally be calculated

$$\begin{aligned} T_{11} &= B_{11}R_{11} + B_{12}R_{21} \simeq -\frac{\eta}{2\delta^2}(\delta - \bar{x}) \left[\frac{1}{2} + \frac{(\delta - \bar{x})^2}{\eta^2}\right] \\ T_{12} &= B_{11}R_{12} \simeq -\frac{\eta}{2\delta^2}(2\delta - \bar{x}) \left[\frac{1}{2} + \frac{(2\delta - \bar{x})^2}{\eta^2}\right] \\ T_{13} &= B_{12}R_{23} \simeq \frac{\eta}{2\delta^2}(\delta - \bar{x}) \left[\frac{1}{2} + \frac{(3\delta - \bar{x})^2}{\eta^2}\right] \\ T_{21} &= B_{21}R_{11} + B_{22}R_{21} \simeq -\frac{3\eta}{4\omega} \left[\frac{1}{2} + \frac{(\delta - \bar{x})^2}{\eta^2}\right] \\ T_{22} &= B_{21}R_{12} \simeq \frac{3\eta}{4\omega} \left[\frac{1}{2} + \frac{(2\delta - \bar{x})^2}{\eta^2}\right] \\ T_{23} &= B_{22}R_{23} \simeq -\frac{\eta}{4\omega} \left[\frac{1}{2} + \frac{(3\delta - \bar{x})^2}{\eta^2}\right]. \end{aligned} \tag{2.34}$$

Using the above results, it is now possibile to calculate $Tr(\underline{\underline{C}})$. In [5], it is shown that $\{\sigma_i\}_{i=1}^4$ can be approximately modeled as a linear function of the distance between the $i-$th AN and the TN with the following expression, given in meters

$$\sigma_i \simeq 0.01 r_i + 0.08. \tag{2.35}$$

The numerical values are derived in [5] using Channel Model 3 described in [39] and the energy detection receiver presented in [11], which is composed of a band-pass filter followed by a square-law device and an integrator, in which the integration interval is set equal to $T_s = 1$ s. We assume that the same energy detector receiver is used here.

Since we consider the four ANs nearest to the TN (so that all $\{r_i\}_{i=1}^4$ are similar), it is expected that the standard deviations $\{\sigma_i\}_{i=1}^4$ of the range estimation errors associated with the four closest ANs are similar. Under the assumption that $\sigma_i \simeq \sigma$, (2.7) reduces to

$$\underline{\underline{Q}} \simeq \sigma^2 \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

Also, (2.24) becomes

$$\begin{aligned} Tr(\underline{\underline{C}}) \simeq \sigma^2 [&(T_{11}+T_{12}+T_{13})^2 + (T_{21}+T_{22}+T_{23})^2 \\ &+ T_{11}^2 + T_{12}^2 + T_{13}^2 + T_{21}^2 + T_{22}^2 + T_{23}^2]. \end{aligned} \tag{2.36}$$

From (2.34) one finds that

$$\begin{aligned} T_{11}+T_{12}+T_{13} &= B_{11}(R_{11}+R_{12}) + B_{12}(R_{21}+R_{23}) \\ T_{21}+T_{22}+T_{23} &= B_{21}(R_{11}+R_{12}) + B_{22}(R_{21}+R_{23}). \end{aligned}$$

Substituting (2.32) and (2.33) into the above equations gives

$$\begin{aligned} T_{11}+T_{12}+T_{13} &\simeq -\frac{(2\delta-\bar{x})\eta}{2\delta^2}\left[\frac{1}{2}+\frac{\vec{x}^2}{\eta^2}\right] \\ T_{21}+T_{22}+T_{23} &\simeq -\frac{\eta}{4\omega}\left[\frac{1}{2}+\frac{\vec{x}^2}{\eta^2}\right]. \end{aligned} \tag{2.37}$$

Finally substituting (2.34) and (2.37) into (2.36), after some simple algebraic manipulations one obtains

$$Tr(\underline{\underline{C}}) \simeq \frac{\sigma^2}{4\omega^2\eta^2\delta^4}\sum_{i=0}^{6} C_i \bar{x}^i \tag{2.38}$$

where

$$\begin{aligned}
C_0 &= \frac{5}{2}\delta^2\eta^4\omega^2 + 26\delta^4\eta^2\omega^2 + 146\delta^6\omega^2 + \frac{5}{4}\delta^4\eta^4 + \frac{27}{2}\delta^6\eta^2 + \frac{117}{2}\delta^8 \\
C_1 &= -\left(468\delta^5\omega^2 + 60\delta^3\eta^2\omega^2 + 3\delta\eta^4\omega^2 + 15\delta^5\eta^2 + 108\delta^7\right) \\
C_2 &= \eta^4\omega^2 + 56\delta^2\eta^2\omega^2 + 606\delta^4\omega^2 + 5\delta^4\eta^2 + 81\delta^6 \\
C_3 &= -\left(24\delta\eta^2\omega^2 + 408\delta^3\omega^2 + 30\delta^5\right) \\
C_4 &= 5\delta^4 + 158\delta^2\omega^2 + 4\eta^2\omega^2 \\
C_5 &= -36\delta\omega^2 \\
C_6 &= 4\omega^2.
\end{aligned}$$

Observe that the coefficients $\{C_i\}_{i=0}^{6}$ depend on the position $\bar{x}$ of the TN, on δ and on η (and, hence, on the width ω of the corridor and on the height ζ of the ANs).

Because of the geometry of the considered scenario, $Tr(\underline{\underline{C}})$ is a periodic function of $\bar{x}$. Therefore, in order to compute its average value, it is sufficient to evaluate it over a period, i.e., for $\delta \leq \bar{x} \leq 2\delta$. The average value of $Tr(\underline{\underline{C}})$ over $\bar{x}$ is given by

$$\mu(\delta) \triangleq \frac{1}{\delta}\int_{\delta}^{2\delta} Tr(\underline{\underline{C}})\,\mathrm{d}\bar{x}. \tag{2.39}$$

In order to optimize the placement of ANs, our strategy consists in finding the distance δ between consecutive ANs that minimizes the average MSE $\mu(\delta)$ of the TN position estimates.

Inserting (2.38) into (2.39) and using the explicit expressions of the coefficients $\{C_i\}_{i=0}^{6}$ in (2.38), the following expression for $\mu(\delta)$ can be obtained

$$\mu(\delta) \simeq \frac{\sigma^2}{4\omega^2\eta^2}\left[4\delta^4 + \left(\frac{76}{35}\omega^2 + \frac{8}{3}\eta^2\right)\delta^2 + \frac{22}{15}\eta^2\omega^2 + \frac{5}{4}\eta^4 + \frac{1}{3}\eta^4\omega^2\delta^{-2}\right].$$

In order to minimize $\mu(\delta)$, we set its derivative to zero. Observing that

$$\frac{\mathrm{d}\mu}{\mathrm{d}\delta} \simeq \frac{\sigma^2}{\omega^2\eta^2}\left(4\delta^3 + \left(\frac{38}{35}\omega^2 + \frac{4}{3}\eta^2\right)\delta - \frac{1}{6}\eta^4\omega^2\delta^{-3}\right) \tag{2.40}$$

if we multiply both sides of the equation by $\delta^3/(4\eta^6)$ and define

$$\beta \triangleq \delta^2/\eta^2 \tag{2.41}$$

it can be concluded that

$$\frac{\mathrm{d}\mu}{\mathrm{d}\delta} = 0 \tilde{\Longleftrightarrow} g(\beta) \triangleq \beta^3 + \left(\frac{19}{70}\lambda^2 + \frac{1}{3}\right)\beta^2 - \frac{1}{24}\lambda^2 = 0 \tag{2.42}$$

where $\lambda = \omega/\eta$ and the approximation sign on top of the equivalence is due to the fact that the expression on the right hand side of (2.40) is approximated.

The cubic equation on the right hand side of (2.42) can be solved analytically. More precisely, defining $b \triangleq 19\lambda^2/70 + 1/3$, $d \triangleq -\lambda^2/24$ and $\Lambda \triangleq d^2/4 + b^3 d/27$, one explicit root of (2.42) is

$$\beta_1 = \sqrt[3]{-\frac{b^3}{27} - \frac{d}{2} + \sqrt{\Lambda}} + \sqrt[3]{-\frac{b^3}{27} - \frac{d}{2} - \sqrt{\Lambda}} - \frac{b}{3}. \tag{2.43}$$

If $\Lambda > 0$, then β_1 is the only real solution of (2.42) and, therefore, the solution is $\beta^* = \beta_1$. If $\Lambda \leq 0$, then all the solutions of (2.42) are real. In this case, one can calculate the remaining two solutions β_2 and β_3 of (2.42) as the solutions of the quadratic equation obtained by dividing $g(\beta)$ by $\beta - \beta_1$, and then select β^* from $\{\beta_1, \beta_2, \beta_3\}$. Note that, by definition (2.41), β^* has to be positive, and, for the considered problem, only one of the solutions $\{\beta_i\}_{i=1}^3$ is positive because the derivative of $g(\beta)$ is $\beta(3\beta + 2b)$ which (since $b > 0$) is always positive for $\beta > 0$ and since $g(0) = d < 0$, so $g(\beta) = 0$ has only one positive solution. Therefore, we can conclude that $\beta^* = \max\{\beta_1, \beta_2, \beta_3\}$.

Finally, from (2.41) the optimal value of δ (denoted as δ^*) is given by

$$\delta^* = \sqrt{\beta^*}\eta = \sqrt{\beta^*(\omega^2 + 4\zeta^2)}. \tag{2.44}$$

As observed in [40], if $\zeta = 0$, i.e., the ANs are located on the same plane where the TN moves, the optimal value δ^* is approximately half of the width of the corridor ($\delta^* \simeq \omega/2$).

	δ^* [m]			
	$\omega = 2$ m	$\omega = 3$ m	$\omega = 4$ m	$\omega = 5$ m
$\zeta = 0$ m	0.95	1.42	1.89	2.37
$\zeta = 1$ m	1.19	1.61	2.04	2.49
$\zeta = 2$ m	1.60	2.00	2.40	2.80
$\zeta = 3$ m	1.96	2.39	2.80	3.19
$\zeta = 4$ m	2.27	2.76	3.19	3.60
$\zeta = 5$ m	2.56	3.10	3.56	3.99
$\zeta = 6$ m	2.81	3.41	3.91	4.36
$\zeta = 7$ m	3.05	3.70	4.24	4.72
$\zeta = 8$ m	3.27	3.97	4.54	5.05
$\zeta = 9$ m	3.48	4.22	4.83	5.38
$\zeta = 10$ m	3.67	4.46	5.11	5.68

Table 2.1: Optimal values of δ predicted by (2.44) as the height ζ of the ANs varies between 0 m and 10 m, and the width ω of the corridor varies between 2 m and 5 m.

2.4 Simulation-based Validation

The closed-form expression for δ^* (2.44) is now validated by simulations. The localization performance is evaluated in terms of RMSE, which is defined as

$$\text{RMSE} \triangleq \sqrt{\mathbb{E}[(\hat{x}-x)^2]+\mathbb{E}[(\hat{y}-y)^2]}.$$

Simulation results are obtained by means of a Matlab simulator, based on the one described in [5], in which the standard IEEE 802.15.4a is implemented using Channel Model 3 described in [39] and the energy detection receiver proposed in [11]. We remark that the simulator does not make the simplifying assumption in (2.28) needed in the formulation of the analytical framework.

Table 2.1 shows the optimal values, predicted by (2.44), of the inter-AN distance, as the width ω of the corridor varies between 2 m and 5 m, and the height ζ of the ANs varies between 0 m and 10 m. Observe that the optimal values δ^* do not satisfy the condition (2.29) behind the approximate expressions (2.28) for all cases. More precisely, the condition (2.29) is fulfilled only for small values of ω and large

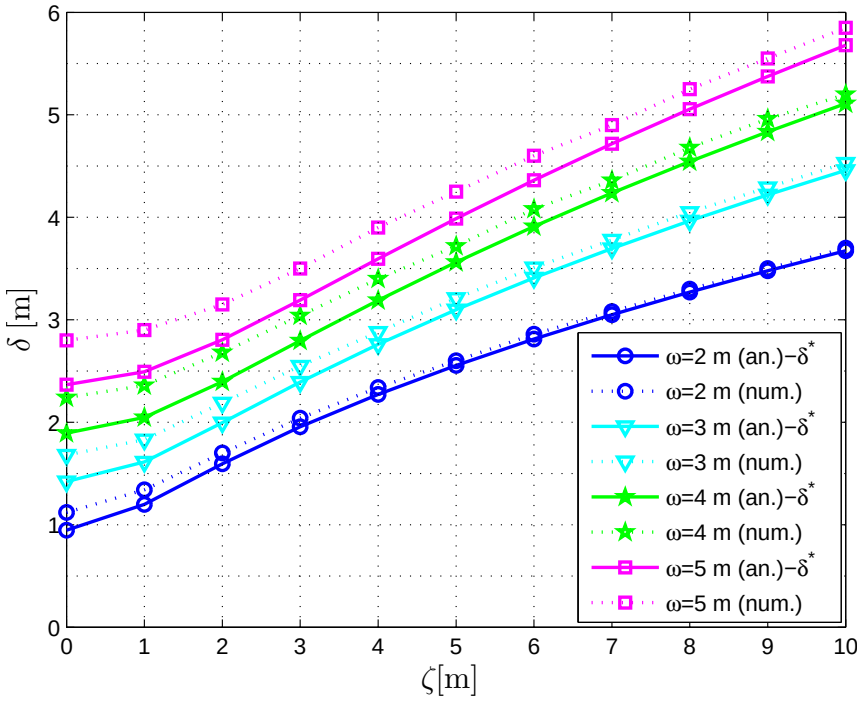


Figure 2.2: Optimal inter-AN distance δ^* as functions of ζ: the values obtained using the closed-form expression (2.44) (*solid lines*) are compared with those obtained numerically (*dotted lines*). Various values of ω are considered.

values of ζ. This is consistent with the assumption of large scenarios, where ANs are attached to the ceiling, which is assumed high.

We remark that, without the Taylor series approximation (2.28), the values of δ^* can still be evaluated numerically. In Figure 2.2, we compare the optimal values δ^* predicted by (2.44), as a function of ζ, with the values obtained numerically with no approximation. Various values of ω are considered. It can be observed that, whenever the Taylor approximation holds (i.e., large ζ and/or small ω), the values of δ^* predicted by (2.44) are very accurate. The largest difference between the closed-form solution and the numerical solution is approximately 0.4 m when $\omega = 5$ m and $\zeta = 0$ m, a case which is not consistent with the large scenario assumption.

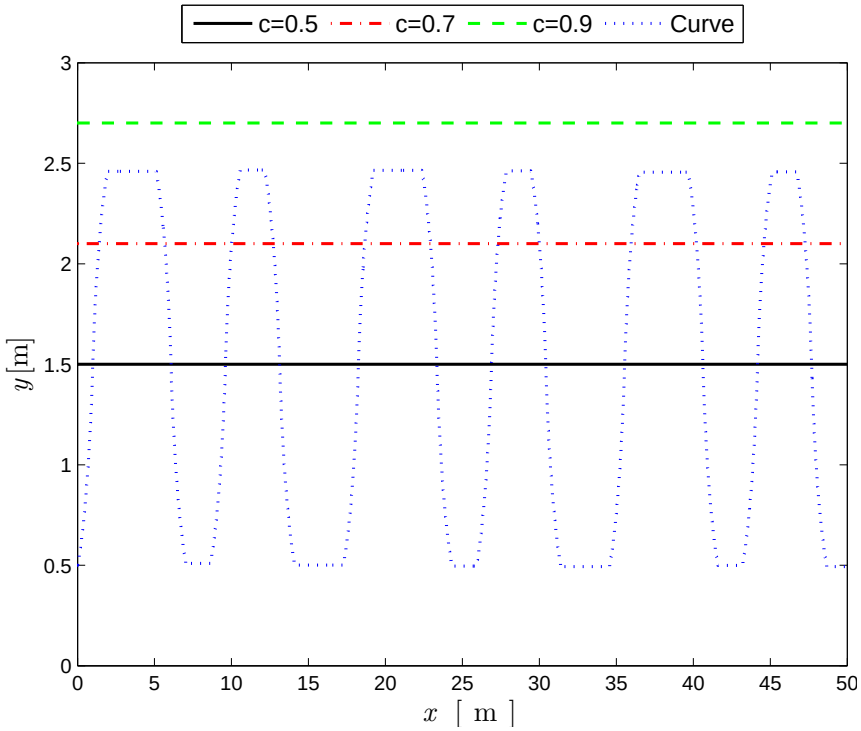


Figure 2.3: Considered paths of the TN on the *xy*-plane.

We now investigate the applicability of the optimal inter-AN distance δ^* predicted by (2.44) to different paths. The RMSE of the TN position estimates is obtained by averaging the results over 100 realizations. Besides considering the scenario in Section 2.3 , we also consider other paths, namely: straight paths not in the middle of the corridor and a non-straight path. More precisely, if the TN moves along a straight line parallel to the walls of the corridor, its position at time t can be expressed as $[x(t), c\,\omega, 0]^T$, where $c \in (0,1)$. The case in which the straight path is exactly in the middle of the corridor corresponds to $c = 0.5$.

First, as shown in Figure 2.3, we consider a scenario where a TN moves in a 50 m-long corridor with width $\omega = 3$ m. Four different paths are considered. More precisely, the solid line corresponds to the case where $c = 0.5$ (the TN moves along

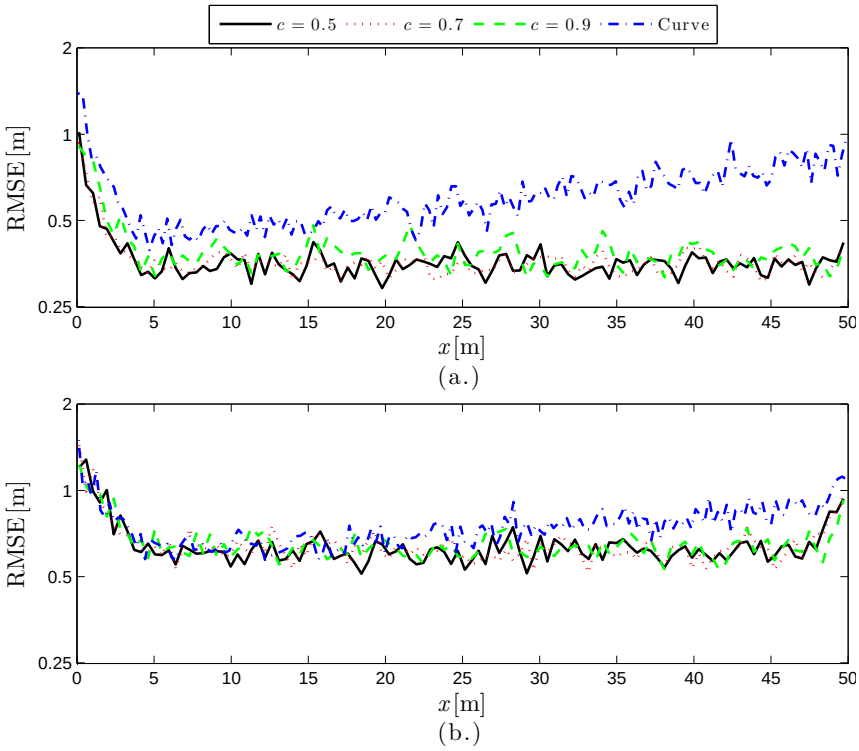


Figure 2.4: For each of the four paths in Figure 2.3, the RMSE of the TN position estimates is shown as a function of the traveled distance, when: (a.) $\zeta = 5$ m and (b.) $\zeta = 8$ m. In all cases, $\omega = 3$ m and $\delta = \delta^*$.

the middle line of the corridor); the dash-dotted line corresponds to the case where $c = 0.7$ (the TN moves along a straight line 0.6 m away from the middle line); the dashed line corresponds to the case where $c = 0.9$ (the TN moves along a straight line 1.2 m away from the middle line, i.e., 0.3 m from a wall). Finally, the dotted line corresponds to the considered non-straight path.

In Figure 2.4, the RMSE of the TN position estimates is shown as a function of the travelled distance, as the TN moves along each of the four paths for two different values of the height ζ of the ANs: (a.) 5 m and (b.) 8 m. From Table 2.1, the optimal value δ^* is 3.10 m for case (a.) and 3.97 m for case (b.). It can be observed that the use

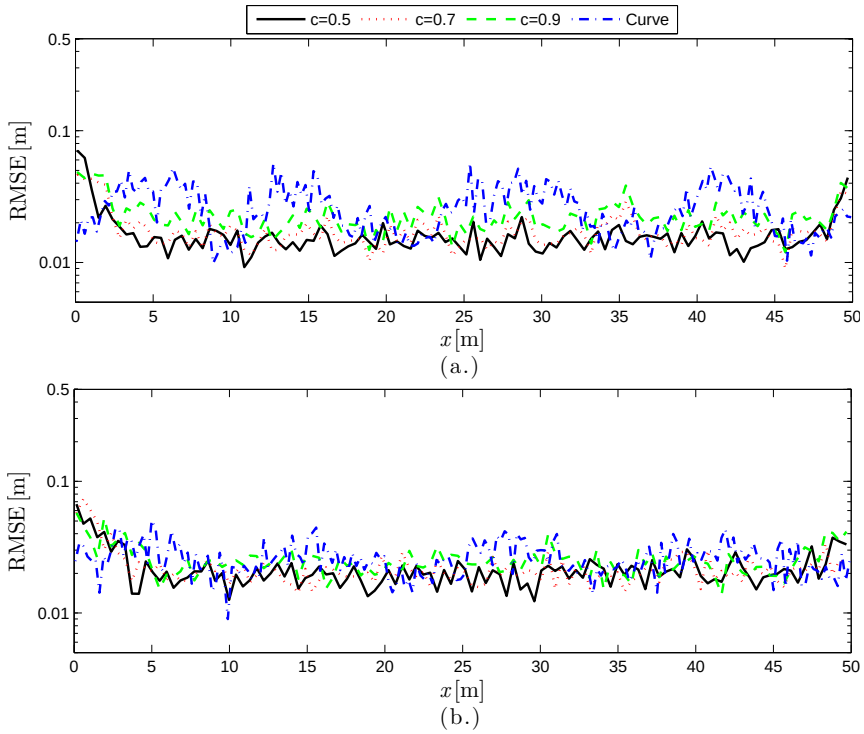


Figure 2.5: For each of the four paths in Figure 2.3, the RMSE of the TN position estimates is shown as a function of the traveled distance, when: (a.) $\zeta = 3$ m and (b.) $\zeta = 6$ m. In all cases, $\omega = 5$ m and $\delta = \delta^*$.

of the optimal value δ^* is effective for all considered three straight paths: the RMSE curves for the three cases are very close to each other. In the case of the non-straight path, the RMSE curve obtained with δ^* is similar to those of the straight paths for $\zeta = 8$ m (b.), while it is higher for $\zeta = 5$ m (a.).

In Figure 2.5, the RMSE is shown, as a function of the TN travelled distance, when $\omega = 5$ m and for two values of the height ζ of the ANs, namely: (a.) 3 m and (b.) 6 m. The inter-AN distance is $\delta = \delta^*$, namely from Table 2.1, $\delta = 3.19$ m and $\delta = 4.36$ m. Observe that, since $\omega = 5$ m, the value $c = 0.7$ corresponds to a straight line 1 m from the middle line, while the value $c = 0.9$ corresponds to a straight line

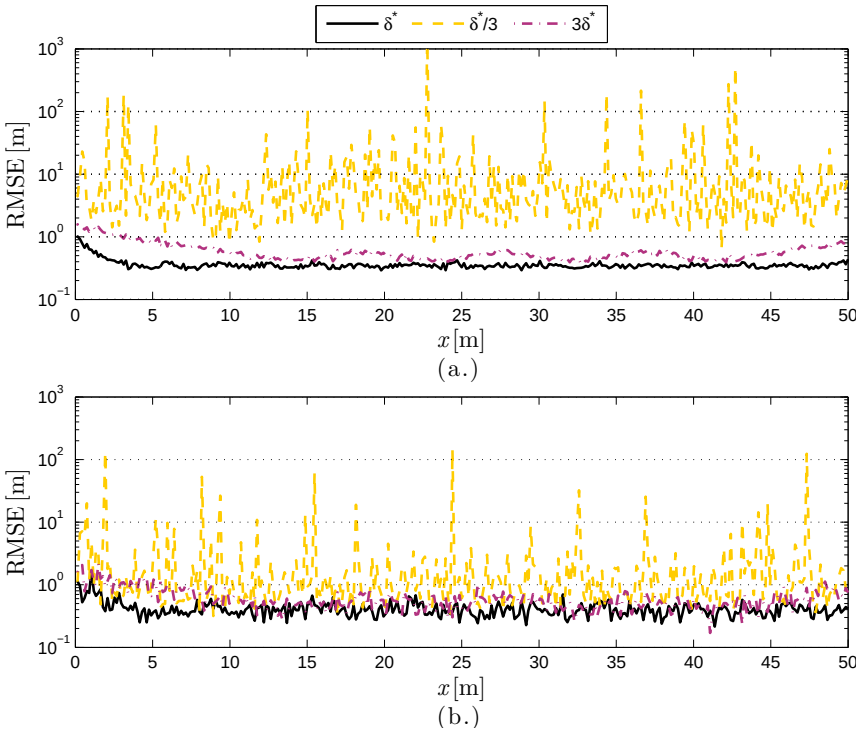


Figure 2.6: RMSE of TN position estimates when the TN moves along a straight line in the middle of the corridor when $\omega = 3$ m: (a.) $\zeta = 5$ m and (b.) $\zeta = 8$ m.

2 m from the middle line, thus half a meter from the wall. The considered non-straight path, instead, is the same as the one in Figure 2.3.

Figure 2.4 and Figure 2.5 show that moving the TN along a path which is not the middle line does not have a significant impact on the RMSE, especially when the height ζ of the ANs is high. In all cases, the RMSE is higher at the beginning and at the end of the corridor. This is because, in such situations, the distances between the TN and the ANs do not comply with the assumption of Section 2.3.

It is now of interest to investigate the effect of a non-optimal AN placement, i.e. when the actual inter-AN distance δ is not δ^*, on the localization performance. In Figure 2.6, we investigate it when $\delta \neq \delta^*$ assuming that the TN moves along a

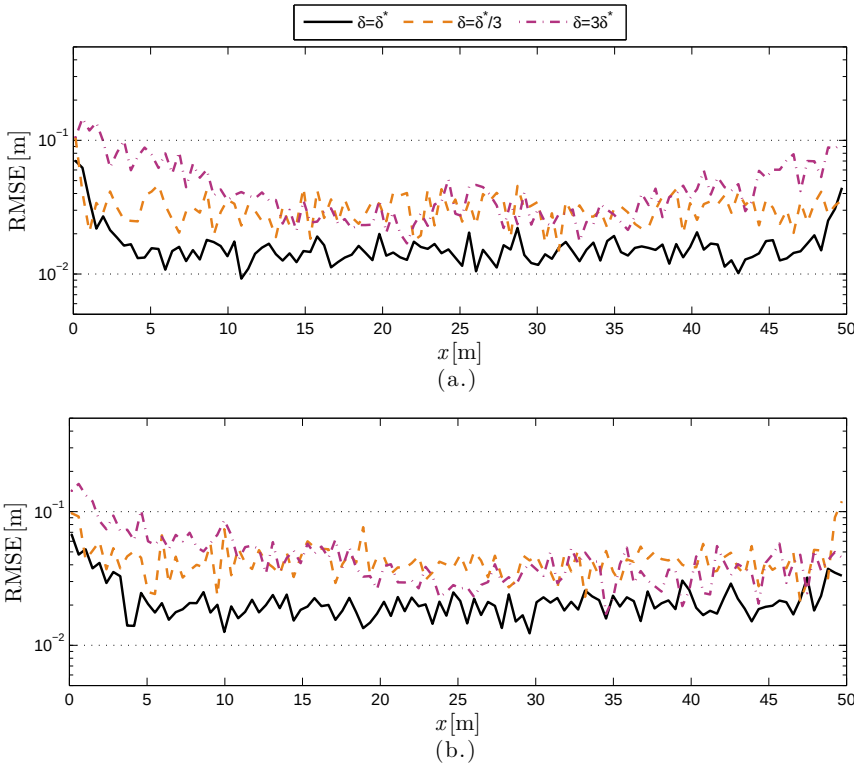


Figure 2.7: RMSE of TN position estimates when the TN moves along a straight line in the middle of the corridor when $\omega = 5$ m: (a.) $\zeta = 3$ m and (b.) $\zeta = 6$ m.

straight line in the middle of the corridor, for (a.) $\zeta = 5$ m and (b.) $\zeta = 8$ m. In both cases, the considered values of δ are: δ^*, $\delta^*/3$, and $3\delta^*$. From Figure 2.6, it can be observed that, as expected, the lowest RMSE curve is obtained when the optimal value δ^* is used. Also, a smaller value of δ leads to significant peaks in the RMSE curve. On the other hand, increasing the value of δ beyond δ^* has a less significant impact on the RMSE curve.

In Figure 2.7, the RMSE is shown assuming that the TN moves in the middle of the corridor and the width of the corridor is set to $\omega = 5$ m. The values $\zeta = 3$ m (a.) and $\zeta = 6$ m (b.) are considered, for different values of δ, namely: δ^*, $\delta^*/3$, $3\delta^*$.

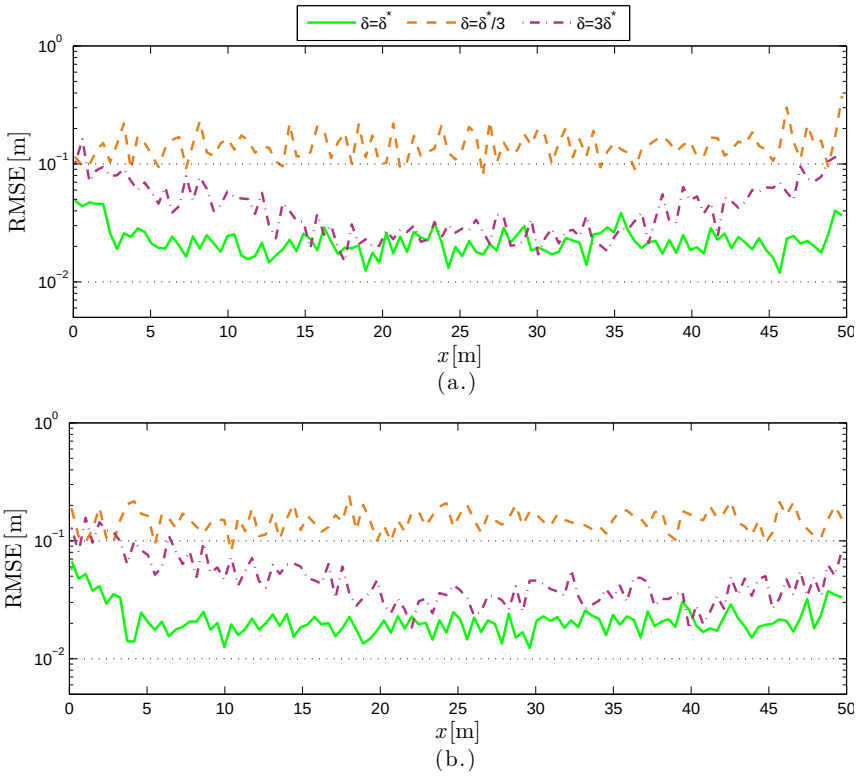


Figure 2.8: RMSE of TN position estimates when the TN moves along a straight line near a wall of the corridor when $\omega = 5$ m: (a.) $\zeta = 3$ m and (b.) $\zeta = 6$ m. In both cases, various values of δ (namely δ^*, $\delta^*/3$, $3\delta^*$) are considered.

In both cases, the RMSE is lower in correspondence to δ^*. It is also significant to observe that, as in the case with $\omega = 3$ m, considering a denser placement of ANs ($\delta < \delta^*$) leads to a performance degradation, with respect to the case with δ^*.

In Figure 2.8, the RMSE is shown assuming that the TN moves, along a corridor of width $\omega = 5$ m, 2 m from the middle line, thus 0.5 m far from the wall, when $\zeta = 3$ m (a.) and $\zeta = 6$ m (b.), for the same values of δ as in Figure 2.8. It can be noticed that, even if the assumption of the TN moving exactly in the middle of a corridor is not verified, the lower RMSE is obtained when the value of δ is set equal to δ^*.

	δ^* [m]			
	$\omega = 2$ m	$\omega = 3$ m	$\omega = 4$ m	$\omega = 5$ m
$\zeta = 0$ m	1.22	1.82	2.43	3.04
$\zeta = 1$ m	1.41	1.98	2.55	3.14
$\zeta = 2$ m	1.74	2.28	2.83	3.39
$\zeta = 3$ m	2.04	2.61	3.15	3.70
$\zeta = 4$ m	2.31	2.91	3.47	4.02
$\zeta = 5$ m	2.55	3.20	3.78	4.34
$\zeta = 6$ m	2.78	3.46	4.07	4.65
$\zeta = 7$ m	2.99	3.71	4.35	4.95
$\zeta = 8$ m	3.18	3.94	4.61	5.23
$\zeta = 9$ m	3.37	4.17	4.87	5.51
$\zeta = 10$ m	3.54	4.38	5.10	5.77

Table 2.2: Optimal values of δ numerically evaluated using the TSML-TDoA method, as the height ζ of the ANs varies between 0 m and 10 m and the width ω of the corridor varies between 2 m and 5 m.

2.5 Comparison between PI and TSML-TDoA Methods

In this section, we compare the values of δ^* predicted by (2.44) with the values numerically obtained when using the TSML-TDoA method [23] which is described in Section 1.3. Such a comparison is meaningful since the TSML-TDoA method can attain, as shown in [8], the CRLB, but it does not lead to a closed-form expression for δ^*. We remark that the TSML-TDoA method involves the solution of a 3×3 system of equations for each position estimate, while the PI method involves the solution of a 2×2 system of equations and is, therefore, more computationally efficient.

Table 2.2 shows the optimal values of δ numerically obtained with the TSML-TDoA method. As in Table 2.1, the width ω of the corridor varies between 2 m and 5 m, and the height ζ of the ANs varies between 0 m and 10 m. By comparing the results in Table 2.1 with those in Table 2.2, it can be observed that the optimal values for δ are very similar for the two cases, especially for high values of ζ and small values of ω, which correspond to the most practical configurations. Also, the values

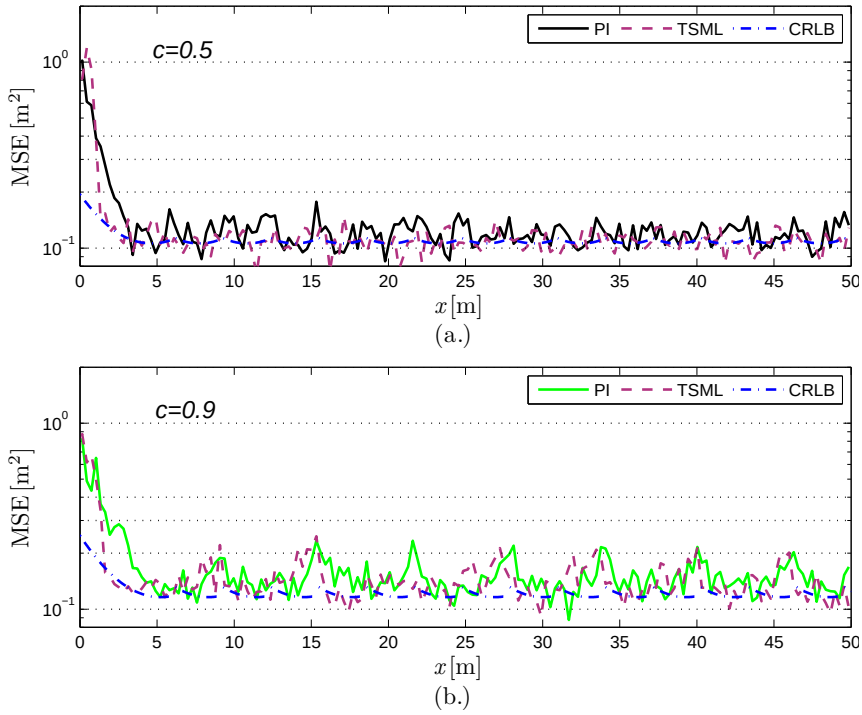


Figure 2.9: MSE of TN position estimates, for the PI method (solid line) and the TSML-TDoA method (dashed line), when the TN moves along the corridor with $\omega = 3$ m and $\zeta = 5$ m, $\delta = \delta^*$ and (a.) $c = 0.5$ (b.) $c = 0.9$. Also the CRLB when $\delta = \delta^*$ is plotted.

obtained with the TSML-TDoA method are very similar to those numerically found for the PI method shown in Figure 2.2.

In Figure 2.9, the MSE is shown as a function of the travelled distance, for both PI and TSML-TDoA methods, when the TN moves along a corridor with width $\omega = 3$ m. More precisely, Figure 2.9 (a.) corresponds to the case in which the TN moves along the middle line of the corridor ($c = 0.5$), while Figure 2.9 (b.) corresponds to the case where the TN moves along a straight line near a wall ($c = 0.9$). In both cases, the height of the ANs $\zeta = 5$ m and the inter-AN distance $\delta = \delta^*$. It can be observed that the PI method and the TSML-TDoA method result in the same localization ac-

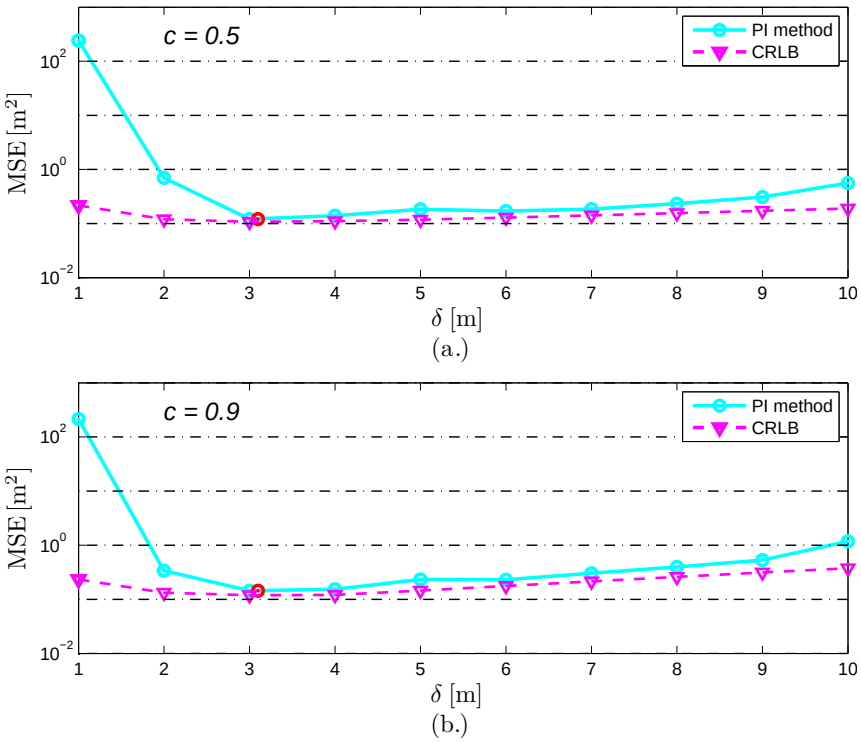


Figure 2.10: Average MSE with PI method (solid line) and average CRLB (dashed line), as a function of δ, when the TN moves along a straight line: (a.) in the middle of a corridor ($c = 0.5$); (b.) near a wall ($c = 0.9$). In both cases, $\omega = 3$ m and $\zeta = 5$ m.

curacy. In Figure 2.9, the CRLB on the MSE evaluated when $\delta = \delta^*$ is also shown. According to [8], the CRLB on the MSE can be expressed as the trace of the matrix $\underline{\underline{G}} \triangleq c^2(\underline{\underline{\Omega}}^T \underline{\underline{Q}}^{-1} \underline{\underline{\Omega}})^{-1}$, where c is the speed of light, $\underline{\underline{Q}}$ is the matrix defined in (2.7) and $\underline{\underline{\Omega}}$ is the following 3×2 matrix

$$\underline{\underline{\Omega}} \triangleq \begin{pmatrix} \frac{x_1 - x}{r_1} - \frac{x_2 - x}{r_2} & \frac{y_1 - y}{r_1} - \frac{y_2 - y}{r_2} \\ \frac{x_1 - x}{r_1} - \frac{x_3 - x}{r_3} & \frac{y_1 - y}{r_1} - \frac{y_3 - y}{r_3} \\ \frac{x_1 - x}{r_1} - \frac{x_4 - x}{r_4} & \frac{y_1 - y}{r_1} - \frac{y_4 - y}{r_4} \end{pmatrix}.$$

From Figure 2.9, the MSE of the PI and the TSML-TDoA methods approaches CRLB except for border effects. It is of interest to investigate the effect when $\delta \neq \delta^*$.

In Figure 2.10, the average MSE is shown as a function of δ when the TN moves along a straigh line (a.) in the middle of the corridor ($c = 0.5$) and (b.) near a wall ($c = 0.9$). In both cases the width of the corridor $\omega = 3$ m and the height of the ANs $\zeta = 5$ m. As expected, the lowest average MSE is reached when $\delta = \delta^*$, the optimal value of δ predicted by (2.44). For comparison purposes, in Figure 2.10 the average CRLB (over the entire path) is also shown, as a function of δ.

Chapter 3

A Swarm Intelligence Approach to Static Localization

Tristis eris si solus eris

– Publius Ovidius Naso

3.1 Introduction

This chapter focuses on the localization of static TNs assuming to know the exact positions of a few ANs. More precisely, we consider a set of static nodes laying on a plane, which may be the floor or the ceiling of an indoor environment, and we assume to know the position of some of them which are considered ANs. The number of ANs should be small in order to reduce costs, but sufficiently large to guarantee reliable position estimates of each TN.

Recall the PI algorithm [59] and the TSML-TDoA algorithm [8] introduced in Chapter 1. We remark that in this chapter we consider bidimensional versions of such algorithms. Both these algorithms deal with solving linear or non-linear systems of equations, whose coefficients contain the coordinates of ANs and the range

estimates between them and the current tag. In some particular cases, for instance, if the considered ANs are on the same line, the matrices involved in the aforementioned algorithms might become ill-conditioned, thus leading to a wrong position estimate of the TN. To avoid this kind of problems, it is possible to re-interpret the geometric localization problem as a minimization problem. More precisely, the initial system of equations of the TSML-TDoA algorithm can be written in terms of an optimization problem and solved through the use of Particle Swarm Optimization (PSO), as in [41]. The PSO algorithm is a soft computing technique introduced in [31]. The idea behind it is to obtain computational intelligence by exploiting simple analogues of social interactions, rather than purely individual cognitive abilities.

In this chapter, various scenarios and different ANs positions in each scenario are considered. The impact of the number of ANs and of distances between nodes is evaluated by simulation. Direct comparison of the performance of geometric algorithms (i.e., the PI or the TSML-TDoA methods) with that of the PSO algorithm shows that the latter can significantly outperform the formers.

A novel improved version of the PSO algorithm, which extends the hybrid approach in [42], is then proposed and its performance is compared with that of the standard PSO algorithm. In particular, the number of iterations that are needed to achieve a given error tolerance in the position estimate is investigated. Simulation results show that the proposed hybrid version of the PSO algorithm guarantees, at a significantly reduced computational cost, faster convergence in all considered scenarios, making it very attractive from an applicative viewpoint.

3.2 Use of PSO in Localization

As anticipated in Section 3.1, all considered static nodes are supposed to be on the same plane, e.g., on the ceiling of a large indoor environment. For this reason, unlike in Chapter 1 and Chapter 2, the coordinates of all the nodes are represented by bidimensional vectors.

A set of M nodes $\{\underline{s}_i\}_{i=1}^{M}$ with known positions, namely the ANs, is used to estimate the position of the remaining nodes with unknown positions, namely the TNs.

All the TNs are estimated in sequence. The true position of the currently considered TN is indicated as $\underline{u} = [x, y]^T$, while the obtained position estimate is denoted as $\hat{\underline{u}} = [\hat{x}, \hat{y}]^T$. Following the notation introduced in (1.3), we assume that the estimated distances can be expressed as a function of the true ones according to (1.4). All the results presented in this chapter are obtained assuming that the distance error has the same statistics defined in (2.3), (2.4), and (2.35).

The starting point for the TSML-TDoA algorithm is system (1.25), where we nowneglect the third column of $\hat{\underline{\underline{G}}}_1$ and the third component of $\hat{\underline{\phi}}_1$ as the considered scenario is bidimensional. Through simple algebraic manipulations, this system can be rewritten as

$$\underline{\underline{B}}\hat{\underline{u}} = \hat{\underline{t}} \tag{3.1}$$

where, using the notation defined in (1.20) and (1.5)

$$\underline{\underline{B}} = -2 \begin{pmatrix} x_{12} & y_{12} \\ x_{13} & y_{13} \\ \vdots & \vdots \\ x_{1M} & y_{1M} \end{pmatrix} \qquad \hat{\underline{t}} = \begin{pmatrix} \hat{r}_2^2 - \hat{r}_1^2 + a_1^2 - a_2^2 \\ \hat{r}_3^2 - \hat{r}_1^2 + a_1^2 - a_3^2 \\ \vdots \\ \hat{r}_M^2 - \hat{r}_1^2 + a_1^2 - a_M^2 \end{pmatrix}. \tag{3.2}$$

Note that with the new formulation (3.1), the measurements affected by noise only appear in vector $\hat{\underline{t}}$, while matrix $\underline{\underline{B}}$ only contains known parameters. Instead of directly solving the system (3.1), we reinterpret it as an optimization problem, with solution $\hat{\underline{u}}$ expressed as

$$\hat{\underline{u}} = \text{argmin}_{\underline{u}} F(\underline{u}) \tag{3.3}$$

where the fitness function $F(\cdot)$ is the following

$$F(\underline{u}) \triangleq ||\hat{\underline{t}} - \underline{\underline{B}}\underline{u}||. \tag{3.4}$$

Among all possible optimization algorithms which could be used to solve (3.1), we now consider the PSO algorithm, originally introduced in [31]. The use of soft computing techniques for the localization of nodes in WSNs has been already proposed in the literature [2, 67].

According to the PSO algorithm, the set of potential solutions of an optimization problem can be modeled as a swarm of particles, whose positions are randomly initialized in the region of interest, and which are guided towards the optimal solution by exploiting social interactions among them. In the following, the size of the swarm is denoted as S. It is assumed that every particle i in the swarm, at each iteration step t ($t \in \{0,1,\ldots,T_{\max}\}$), is associated with a position $\underline{x}^i(t)$ (within the region of interest) and with a velocity $\underline{v}^i(t)$, which are both randomly initialized with values $\underline{x}^i(0)$ and $\underline{v}^i(0)$ and which are updated at each iteration [15]. Moreover, it is supposed that the entire system has a global memory, which allows each particle to know, at every iteration, not only its own best position according to the lowest value of the fitness function (3.4), but also the best position among the ones reached by any other particle in the swarm. Each particle also keeps track of the values of the fitness function in correspondence to both its best position and the global best position. All such values are used to iteratively update the velocity and the position of every particle.

From [61], the update rule for the velocity of particle $i \in \{1,\ldots,S\}$ is

$$\underline{v}^i(t+1) = \omega(t)\underline{v}^i(t) + c_1 R_1(t)(\underline{y}^i(t) - \underline{x}^i(t)) + c_2 R_2(t)(\underline{y}(t) - \underline{x}^i(t)) \tag{3.5}$$

where $\omega(t)$ is a weight called inertial factor, c_1 and c_2 are positive real parameters called cognition parameter and social parameter, $R_1(t)$ and $R_2(t)$ are independent random variables uniformly distributed in $[0,1]$, whose realizations are drawn at each iteration step. Finally, $\underline{y}^i(t)$ is the position of the $i-$th particle with the best objective function and $\underline{y}(t)$ is the position of the particle with the best (among all particles) objective function reached until iteration t [54]. For the considered minimization problem (3.3), such positions can be expressed as follows

$$\underline{y}^{(i)}(t) = \operatorname{argmin}_{\underline{z}\in\{\underline{x}^{(i)}(0),\ldots\underline{x}^{(i)}(t)\}} F(\underline{z}) \tag{3.6}$$

$$\underline{y}(t) = \operatorname{argmin}_{\underline{z}\in\{\underline{y}^{(1)}(t),\ldots\underline{y}^{(S)}(t)\}} F(\underline{z}). \tag{3.7}$$

The rationale behind the update rule (3.5) is to add to the velocity at the previous iteration (which is weighed by a multiplicative factor) a stochastic combination of the direction to the best position of the $i-$th particle and to the best global position. The

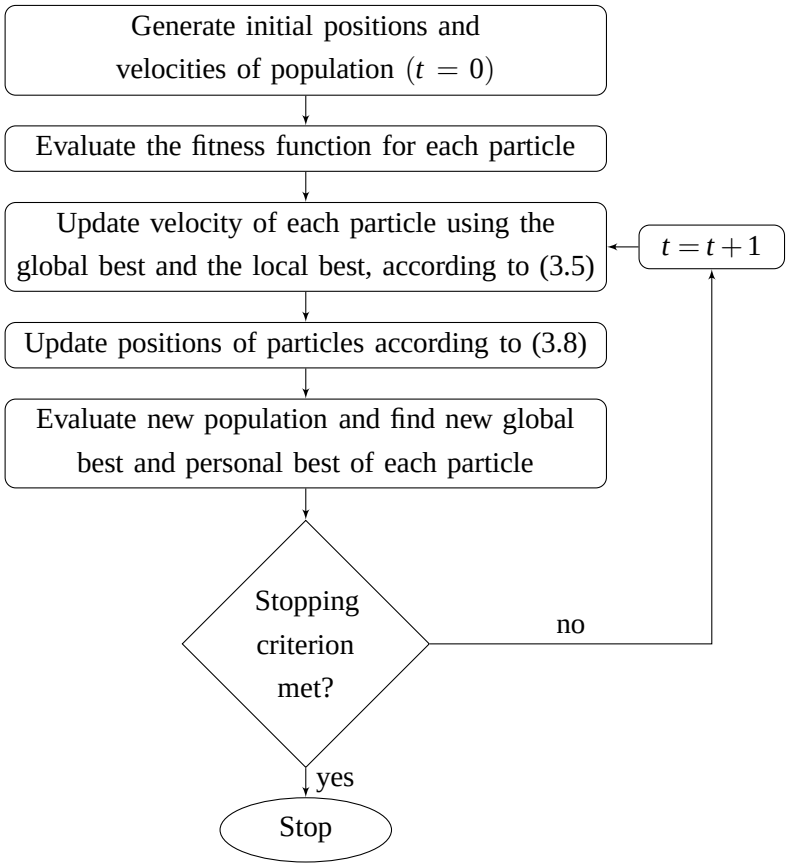


Figure 3.1: Block diagram of the PSO algorithm.

velocity calculated in (3.5) is then added to the current position, in order to update the position of the corresponding particle, according to the rule

$$\underline{x}^i(t+1) = \underline{x}^i(t) + \underline{v}^i(t+1) \qquad i \in \{1,\ldots,S\}. \tag{3.8}$$

This process is iterated until a stopping criterion, such as achieving a satisfactory value of the fitness function or reaching a maximum number of iterations, is met. The solution of (3.3) will then correspond to the position of the particle which best suits the optimization requirements in the last iteration [43]. The steps of the PSO algorithm are summarized in the block diagram in Figure 3.1.

3.3 Comparison with Geometric Methods

In this section, the performance of the swarm intelligence approach described in Section 3.2 is compared with the performance of geometric methods, namely the TSML-TDoA algorithm and the PI algorithm introduced in Subsection 1.3.3 and Subsection 1.3.4, respectively. We compare, through simulations, the performance of the aforementioned algorithms, considering the impact of the number of ANs and of the distance between ANs and TNs.

All the simulation results presented in this section are obtained by a Matlab simulator compliant with the propagation model introduced in Section 2.2, in which we implemented the two geometric localization methods and the PSO algorithm. The performance is evaluated in terms of the RMSE between true and estimated positions. In the following, 100 independent simulation runs per scenario are considered.

In order to obtain the simulation results presented in this section, the PSO algorithm is implemented with a population of $S = 40$ individuals and with a stopping criterion corresponding to the achievement of 50 PSO iterations. We consider $c_1 = c_2 = 2$, in order to make both the averages of the weights for social and cognition contributions equal to 1 [45]. The initial positions of all the particles are randomly initialized over the entire region of interest. The choice of the inertial factor $\omega(t)$ deserves some more comments. From (3.5), it can be observed that this parameter quantifies the ability of the swarm to explore new areas. More precisely, a relatively large value of $\omega(t)$ improves the exploration ability of the swarm, which is a good feature to avoid local optima in global optimization problems. For this reason, the inertial factor $\omega(t)$ is chosen to be a decreasing function of the number of iterations, in order *(i)* to guarantee low dependence of the solution on the initial population and *(ii)* to reduce the exploration ability of the algorithm. This choice makes the method simliar to a local search as the number of iterations increases [61]. In the following simulations, it is assumed that the initial value of the inertial factor is $\omega(0) = 0.9$ and that it decreases linearly to 0.4, reached at the last iteration.

First, we investigate how the obtained localization accuracy changes as the number of ANs reduces. The number of ANs should be as small as possible, in order to

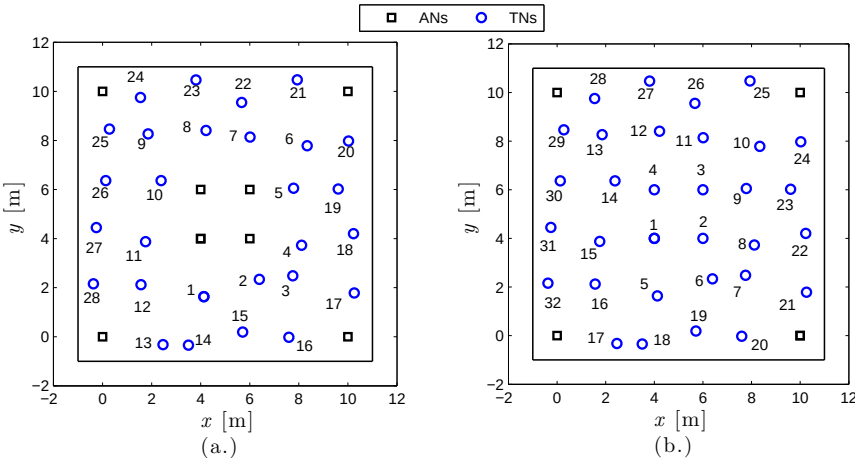


Figure 3.2: Almost uniform distribution of nodes inside a 10 m side square with (a.) 8 ANs and (b.) 4 ANs.

reduce the implementation costs, but also large enough to guarantee accurate estimates for the positions of TNs [47]. Several scenarios with different configurations of ANs are considered.

The first scenario we consider is shown in Figure 3.2, where black squares represent the ANs while blue circles represent the TNs. In this configuration, 36 nodes are almost uniformly placed in the region of interest, namely a square with 10 m-long sides. In particular, in Figure 3.2 (a.) the positions of 8 out of the total number of nodes are assumed to be known, while in Figure 3.2 (b.) the number of ANs is halved, i.e., the positions of only 4 nodes are known. Each TN is associated with a number, denoted as TN index.

In Figure 3.3 the RMSE corresponding to the scenarios in Figure 3.2 is shown as a function of the TN index. In all cases, the RMSE is computed using the three considered algorithms, namely the TSML-TDoA algorithm, the PI algorithm, and the PSO algorithm.

Figure 3.3 (a.) shows the RMSE relative to each TN in the scenario represented in Figure 3.2 (a.), i.e., when the number of ANs is 8. In this case, by comparing the RMSE of the three algorithms, it can be noticed that there are no significant

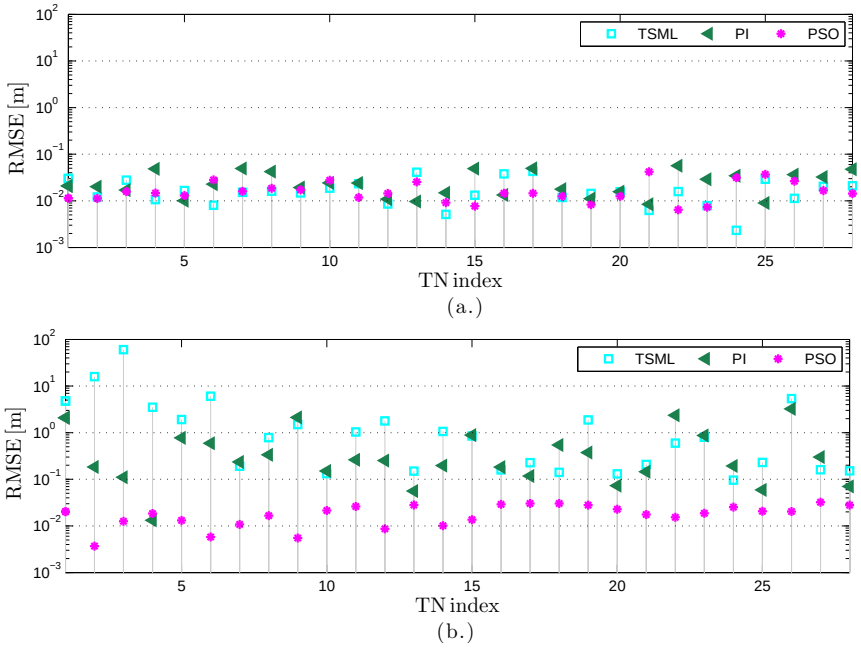


Figure 3.3: RMSE obtained with the TSML-TDoA algorithm (*cyan squares*), the PI algorithm (*green triangles*), and the PSO algorithm (*magenta asterisks*) when considering (a.) the scenario in Figure 3.2 (a.) and (b.) the scenario in Figure 3.2 (b.).

differences in the order of magnitude of the error and the three algorithms guarantee an accurate position estimate of all the TNs. Similarly, in Figure 3.3 (b.) the RMSE corresponding to each TN in the scenario in Figure 3.2 (b.) is shown. In this case, the number of ANs is 4 and they are used to estimate the positions of the remaining 32 TNs. Both the TSML-TDoA and the PI algorithms lead to a far inaccurate localization estimate for many TNs.

On the contrary, the accuracy obtained when using the PSO algorithm is still satisfactory. Moreover, a comparison between Figure 3.3 (a.) and Figure 3.3 (b.) shows that the performance of the PSO algorithm when using only 4 ANs guarantees the same performance obtained when using 8 ANs, as the order of magnitude of the RMSE is the same in both cases.

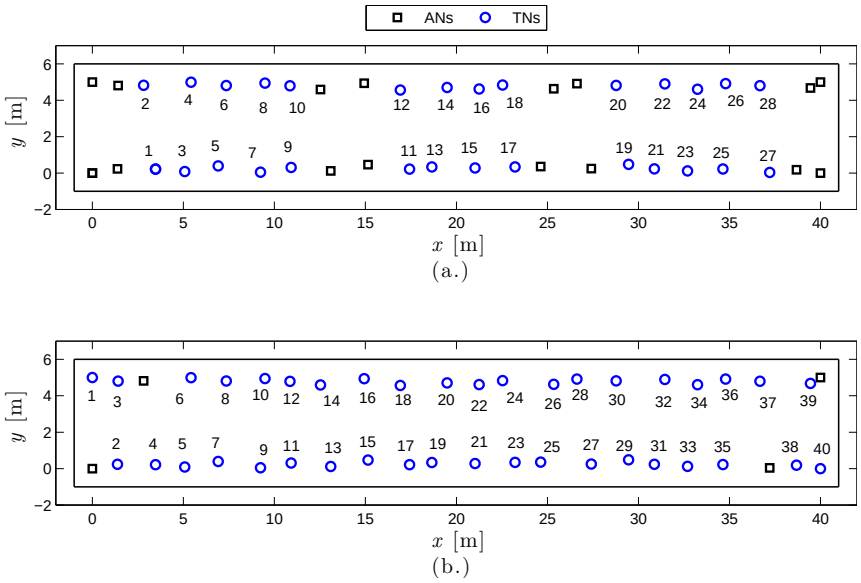


Figure 3.4: Distribution of nodes along a corridor 40 m long and 5 m wide with (a.) 16 ANs and (b.) 4 ANs.

We now consider a different scenario where 44 nodes are located along a corridor 40 m long and 5 m wide, as shown in Figure 3.4, where black squares represent the ANs and blue circles represent the TNs. In particular, in Figure 3.4 (a.) the number of ANs is 16, while in Figure 3.4 (b.) the number of ANs is only 4.

In Figure 3.5 (a.) the RMSE relative to the scenario in Figure 3.4 (a.) is shown as a function of the TN index. A comparison between the RMSE of the three algorithms shows that they guarantee the same performance. On the other hand, Figure 3.5 (b.), relative to the scenario in Figure 3.4 (b.), shows that when halving the number of ANs, the TSML-TDoA algorithm and the PI algorithm lead to inaccurate estimates for many TNs, while the PSO algorithm still guarantees satisfactory results.

It can be concluded that in both scenarios shown in Figure 3.2 and Figure 3.4, the reduction of the number of ANs leads to a significant degradation in the accuracy of the TSML-TDoA algorithm and of the PI algorithm, but it does not influence the performance of the PSO algorithm.

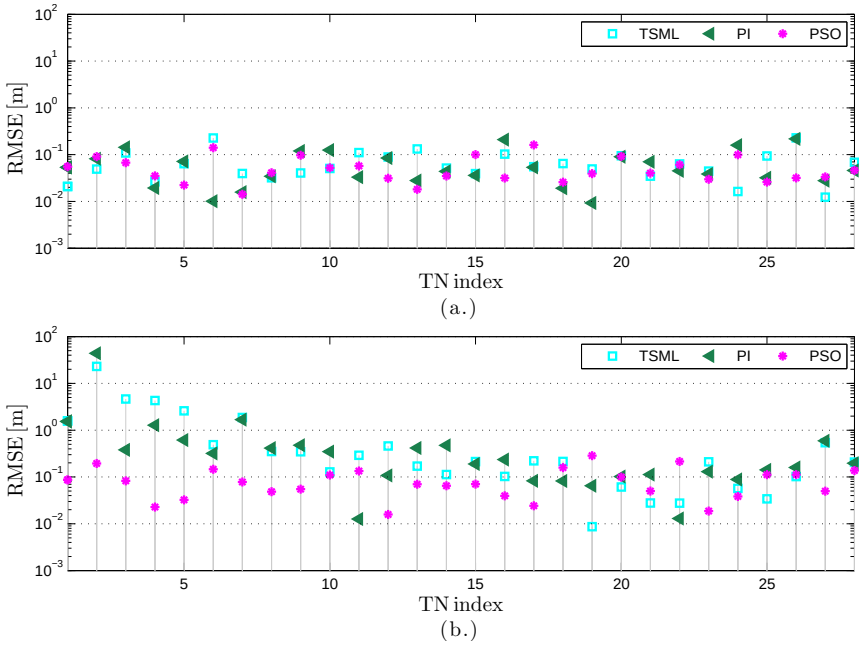


Figure 3.5: RMSE obtained with the TSML-TDoA algorithm (*cyan squares*), the PI algorithm (*green triangles*), and the PSO algorithm (*magenta asterisks*) when considering (a.) the scenario in Figure 3.4 (a.) and (b.) the scenario in Figure 3.4 (b.).

We now introduce a slight modification of the PSO parameters used so far. Recall that the fitness function in the considered minimization problem is expressed in terms of an Euclidean norm. Hence, the fitness function has no local minima, i.e. the minimization problem is convex.

Recalling that large values of ω increase the ability of the swarm to explore new areas of the solution space, one can set $\omega(t) = 0$, as proposed in [69]. This choice, besides accelerating the speed of convergence, also reduces the computational cost of the algorithm. As a matter of fact, since the first addend at the right hand side of (3.5) is no longer considered, there is no need to initialize and compute velocities. The update rule for the positions $\{\underline{x}^i\}_{i=1}^S$ of particles becomes

$$\underline{x}^i(t+1) = \underline{x}^i(t) + c_1 R_1(t)(\underline{y}^i(t) - \underline{x}^i(t)) + c_2 R_2(t)(\underline{y}(t) - \underline{x}^i(t)). \tag{3.9}$$

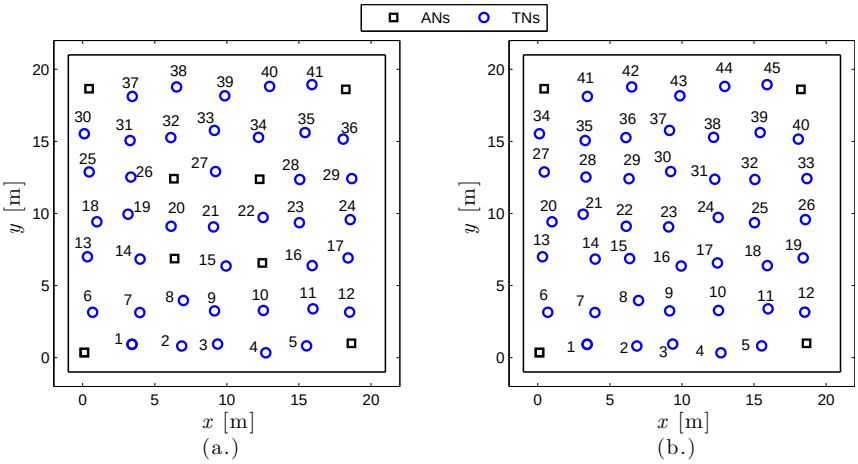


Figure 3.6: Almost uniform distribution of nodes inside a 20 m side square with (a.) 8 ANs and (b.) 4 ANs.

All the simulation results in the remainder of this section are obtained under this assumption, maintaining the same values of all the others parameters involved in the PSO algorithm. Moreover, we now consider different scenarios, where we assume that 49 nodes are placed in a square whose edges are 20 m long. Observe that in these scenarios the density of nodes is lower than that of the previously considered scenarios. Two spatial distributions of the 49 nodes are considered, and for each one of them two configurations of ANs are investigated.

First, we consider the scenarios shown in Figure 3.6, where the nodes are almost uniformly placed in the region of interest. In particular, in Figure 3.6 (a.) the number of ANs is 8: 4 ANs are near the corners of the considered region, while the remaining 4 ANs are near the center of the square. In Figure 3.6 (b.), the number of ANs is halved and only the 4 nodes at the vertices of the square are ANs.

In Figure 3.7, the RMSE corresponding to each node is shown. More precisely, Figure 3.7 (a.) refers to the scenario in Figure 3.6 (a). It can be noticed that, in this case, the TSML-TDoA algorithm, the PI algorithm, and the PSO algorithm guarantee the same performance in the position estimate of each node. On the contrary, when

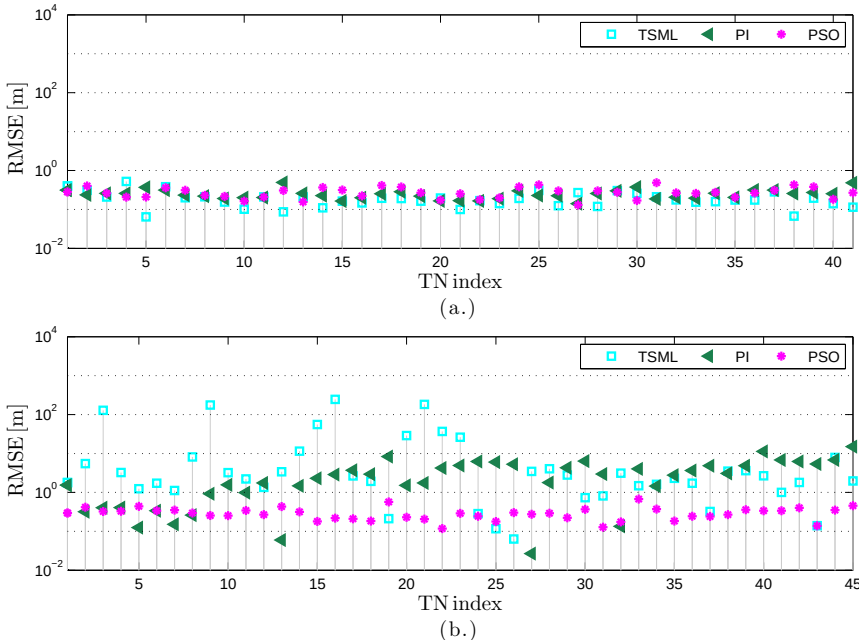


Figure 3.7: RMSE obtained with the TSML-TDoA algorithm (*cyan squares*), the PI algorithm (*green triangles*), and the PSO algorithm (*magenta asterisks*) when considering (a.) the scenario in Figure 3.6 (a.) and (b.) the scenario in Figure 3.6 (b.).

reducing the number of ANs, thus considering the scenario in Figure 3.6 (b.), it can be seen from the results in Figure 3.7 (b.) that the TSML-TDoA algorithm and the PI algorithm do not guarantee a reliable position estimate for the majority of nodes. In particular, the RMSE obtained with the TSML-TDoA algorithm is highly fluctuating (from 10^{-1} m to nearly 10^{3} m). At the opposite, the accuracy of the PSO algorithm is similar for all TNs and it is still good. Moreover, the order of magnitude of the RMSE is analogous to the case with 8 ANs.

In the scenario shown in Figure 3.8, the same number of nodes are randomly placed on a square with the same area of the one in Figure 3.6, namely a square whose edges are 20 m long. As in the previous case, the positions of the nodes in Figure 3.8 (a.) and in Figure 3.8 (b.) are the same, but in Figure 3.8 (a.) the number

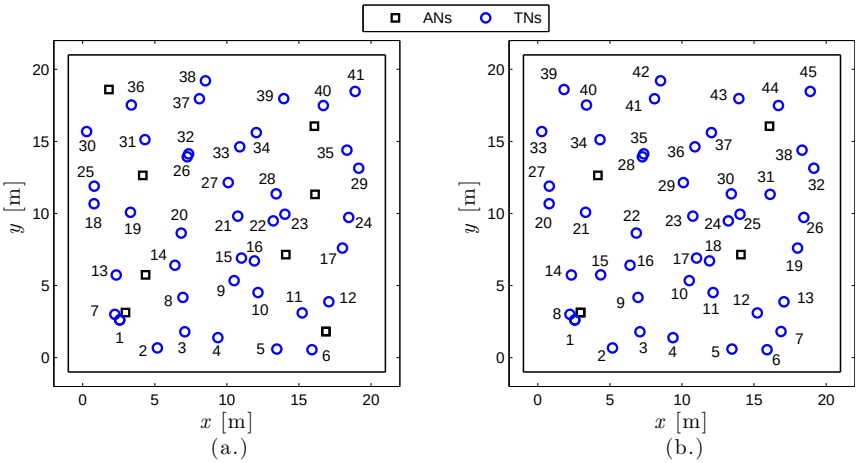


Figure 3.8: Random distribution of nodes inside a 20 m side square with (a.) 8 ANs and (b.) 4 ANs.

of ANs is 8, while in Figure 3.8 (b.) there are only 4 ANs. Note that the ANs have different positions inside the square with respect to the ones in Figure 3.6. In particular, in Figure 3.6 (b.) the 4 ANs are no longer near the corners but they are closer to the center of the square.

Observe that in the configuration shown in Figure 3.8 some of the nodes are very close to each other. For example, in Figure 3.8 (a.) nodes 26 and 32 nearly overlap, and nodes 1 and 7 are very close to one of the ANs. Similarly, in Figure 3.8 (b.) node 28 is very close to node 35, and nodes 1 and 8 are nearby one of the ANs.

Looking at Figure 3.9 (a.) it can be noticed once again that 8 ANs are sufficient to guarantee accurate position estimates for each node, and that there are no significant differences in the performance of the three algorithms. On the other hand, considering the scenario in Figure 3.8 (b.), it can be seen from the results in Figure 3.9 (b.) that halving the number of ANs does not allow the TSML-TDoA algorithm and the PI algorithm to obtain a reliable position estimate for the majority of the nodes, while the PSO algorithm still guarantees the same order of magnitude of the error. In particular, the RMSE of the TSML-TDoA and of the PI methods has significant peaks in

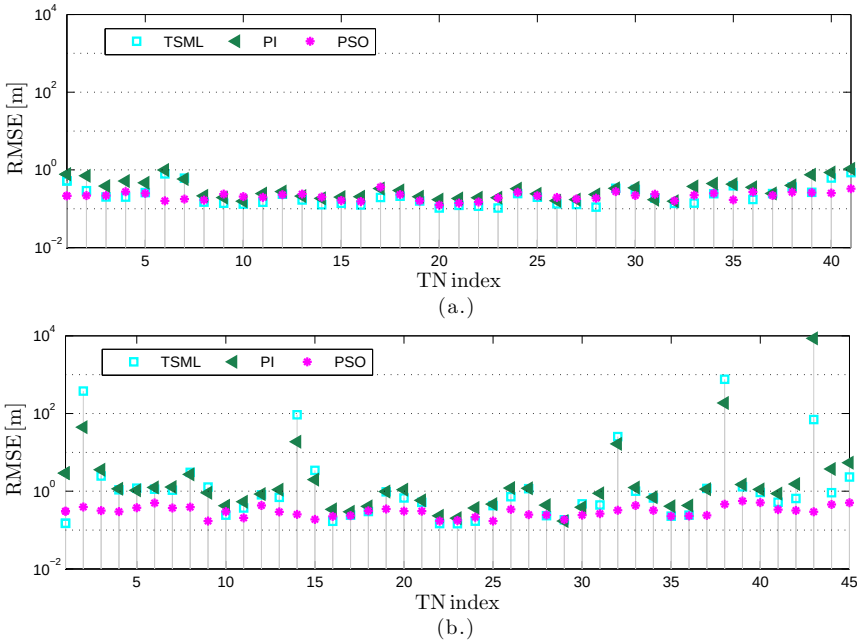


Figure 3.9: RMSE obtained with the TSML-TDoA algorithm (*cyan squares*), the PI algorithm (*green triangles*), and the PSO algorithm (*magenta asterisks*) when considering (a.) the scenario in Figure 3.8 (a.) and (b.) the scenario in Figure 3.8 (b.).

correspondence to some nodes, such as nodes 2 and 14. This makes the use of such algorithms impractical because RMSE peaks are three order of magnitude higher than the average of the remaining values.

A performance degradation when using the TSML-TDoA or the PI algorithm can be noticed from the previous simulations, as the number of ANs is reduced. This phenomenon is due to the fact that a small number of range measurements may not give enough independent data to get accurate position estimates. Mathematically, this means that some of the matrices involved in the solution of the localization problem are ill-conditioned, thus leading to wrong results. At the opposite, the PSO algorithm only involves the evaluation of a function at different points, thus avoiding the solution of systems which may be ill-conditioned.

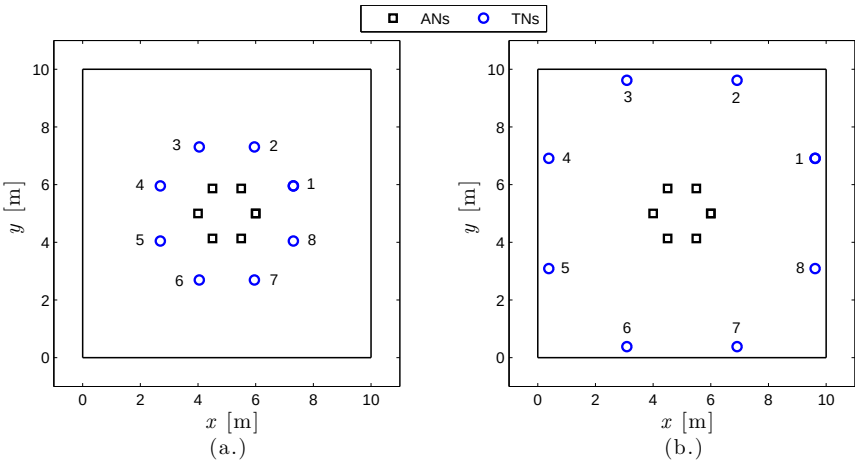


Figure 3.10: Configuration with 6 ANs and 8 TNs laying on circumferences with radius (a.) 2.5 m and (b.) 5 m.

We now investigate the impact of the distance between ANs and TNs on the resulting position estimates. For this purpose, we consider the two scenarios described in Figure 3.10, where 6 ANs (i.e., the 6 black squares closer to the center of the square) are used to estimate the positions of the 8 TNs, denoted as blue circles, which are placed on two circumferences centered in the baricenter of the ANs and with different radii. In particular, in Figure 3.10 (a.) the distance between the TNs and the baricenter of the ANs is 2.5 m, while in Figure 3.10 (b.) the TNs are positioned on a circumference centered in the baricenter of the ANs whose radius is 5 m.

Simulation results in Figure 3.11 show that, in both cases, the PSO algorithm outperforms the TSML-TDoA and PI algorithms by nearly one order of magnitude. Moreover, from a comparison between Figure 3.11 (a.) and Figure 3.11 (b.) it can be observed that, as the distance between ANs and TNs increases, the performance of the TSML-TDoA and PI algorithms significantly degrades. As a matter of fact, the RMSE increases by almost an order of magnitude when the radius of the circumference to which the TNs belong doubles. At the opposite, the accuracy of the PSO algorithm does not change.

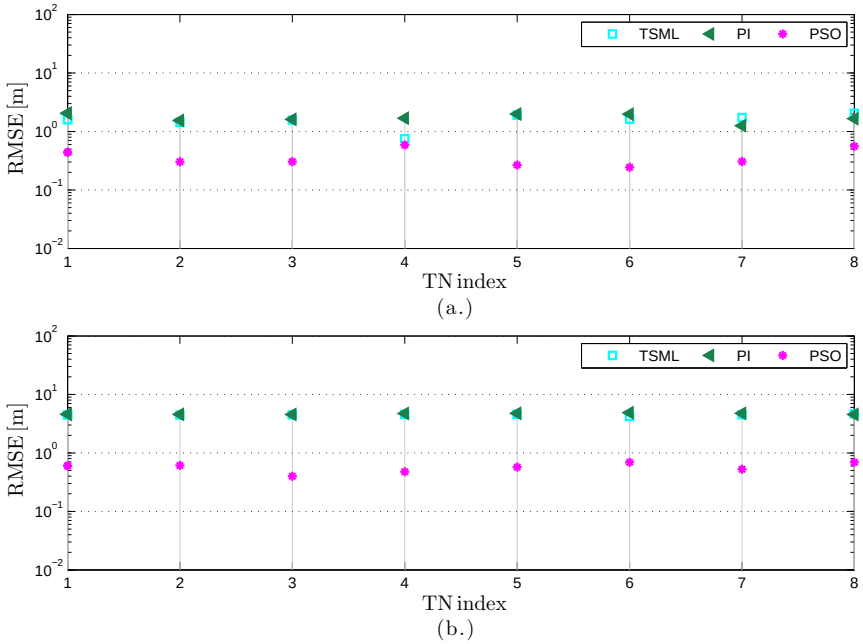


Figure 3.11: RMSE obtained with the TSML-TDoA algorithm (*cyan squares*), the PI algorithm (*green triangles*), and the PSO algorithm (*magenta asterisks*) when considering (a.) the scenario in Figure 3.10 (a.) and (b.) the scenario in Figure 3.10 (b.).

3.4 A Hybrid PSO

Many different versions of the PSO algorithm have been studied in the literature, known as binary particle swarm [32], adaptive particle swarm [71], bare bones particle swarm [30], and hybrid particle swarm [25]. In this section, a hybrid version of the PSO algorithm, denoted as HPSO, is proposed. The main assumptions behind the standard PSO algorithm still hold, but we introduce a new processing step, at each iteration, after the computation of the new estimated positions of the particles and the corresponding values of the fitness function. To be more precise, among all the S particles, we consider the $T \triangleq \lfloor S/2 \rfloor$ ones which have the worst (namely, the higher) values of the fitness function and we move them in a neighbourhood of the particle which has the best (namely, the lowest) value of the fitness function.

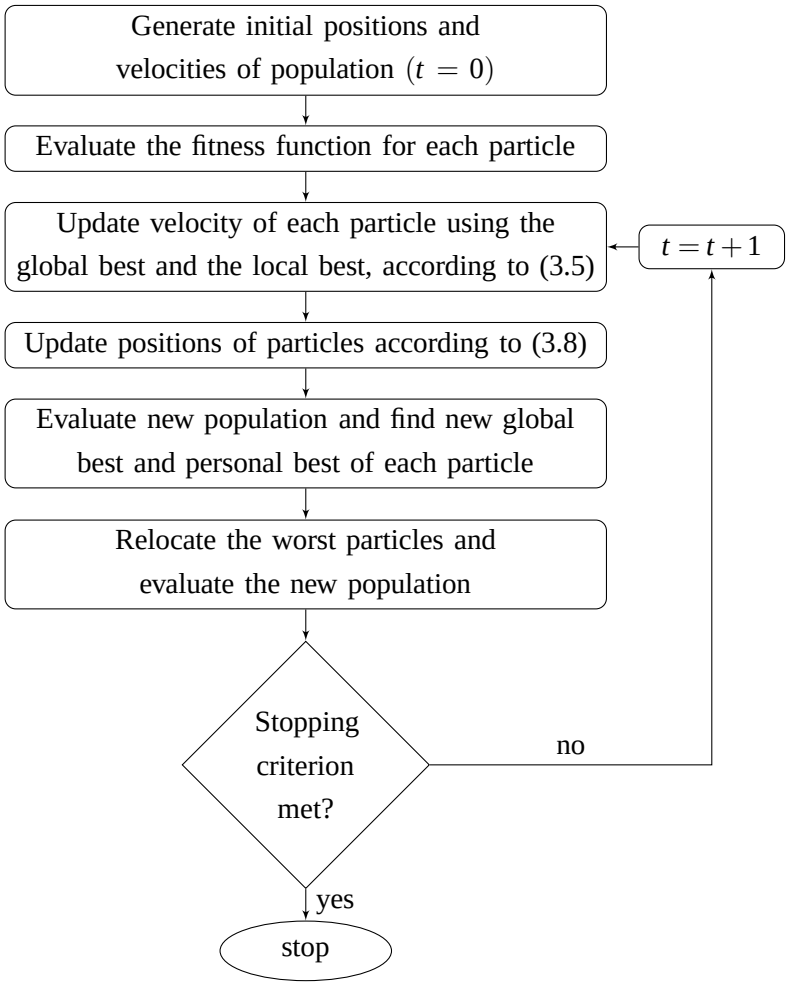


Figure 3.12: Block diagram of the proposed HPSO algorithm

The proposed HPSO algorithm is summarized in the block diagram in Figure 3.12 and it can be described as follows.

At each iteration t of the PSO algorithm, given the current fitness function of each particle, we consider a permutation $\mathscr{P}$ in the set of indices $\{1,\ldots,S\}$, such that

$$F(\underline{x}^{\mathscr{P}[i]}(t)) \leq F(\underline{x}^{\mathscr{P}[j]}(t)) \quad \text{if} \quad i \leq j.$$

In other words, we order the indices of all particles in ascending order, starting form the one with the best fitness value (whose position is $\underline{x}^{\mathscr{P}[1]}$) to the one with the worst value of the fitness function (whose position is $\underline{x}^{\mathscr{P}[S]}$).

Then, for every index $k \in \{1, \ldots, T\}$, we replace the positions of the particles with indices $\{\mathscr{P}[T+k]\}$, (i.e., the worst ones) with positions close to the particles with smaller values of the fitness function. In Section 3.5 two different rules for this relocation of particles are proposed and they are applied in different localization scenarios. After this relocation, the fitness function of the T just moved particles is evaluated, and, from then, the proposed algorithm continues following the steps of the standard PSO algorithm.

The changes introduced in the hybrid version of the PSO algorithm are meant to speed up the convergence to the optimal solution. As a matter of fact, the effect of the additional step is to rapidly push the swarm towards the (currently) best solution. We remark that this idea is effective when the PSO algorithm is applied to the considered localization problem because the fitness function in (3.4) is convex and, therefore, the minimization problem described in (3.3) has no local minima. Generally speaking, since the PSO algorithm is usually applied to global optimization problems, the effect of the proposed modification could make the algorithm converge to a local minima, thus leading to a suboptimal solution.

3.5 Comparison between PSO and HPSO

It is worth observing that the introduction of the additional step in Figure 3.12 implies that the computational cost of each iteration of the HPSO algorithm is approximately 1.5 times the cost of each iteration of the PSO algorithm because of the relocation and the revaluation of the fitness function of half of the particles. However, in this section it will be shown that, from a global computational viewpoint, the HPSO algorithm turns out to be significantly more efficient than the PSO algorithm.

In order to evaluate the convergence speed, instead of considering, as a stopping criterion, a fixed number of iterations (as in Section 3.3), we now consider the achievement of a fixed (maximum) RMSE. Two tolerance values for the RMSE are

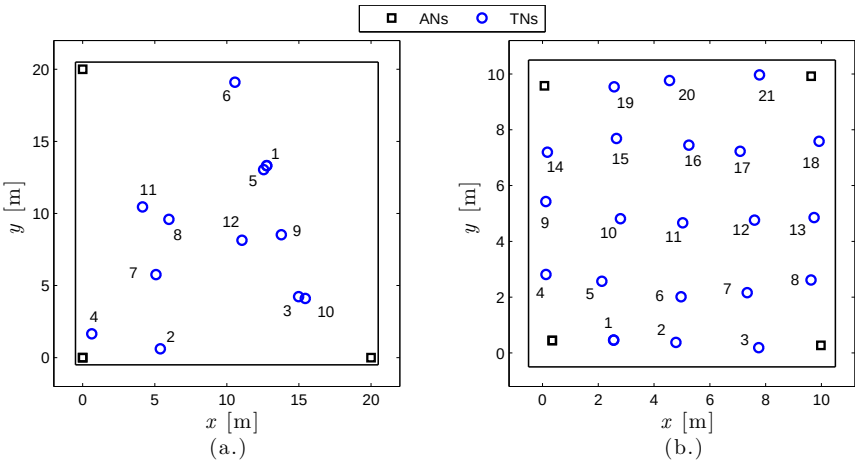


Figure 3.13: Two possible configurations of nodes inside a 10 m side square with (a.) random distribution and (b.) almost uniform distribution.

considered, namely 10^{-1} m and $5 \cdot 10^{-2}$ m. In particular, in Section 3.5.1 we investigate the impact of the number of iterations, whereas in Section 3.5.2 the impact of the swarm size is investigated. Finally, in Section 3.5.3 a comparison of the computational costs of the PSO and the HPSO algorithms is presented.

The simulation results that follows are relative to the scenarios in Figure 3.13. When considering the scenario in Figure 3.13 (a.) the relocation rule for the particles with indices $\{\mathscr{P}[T+k]\}_{k=1}^{T}$ is the following

$$\underline{x}^{\mathscr{P}[T+k]}(t) \leftarrow \underline{x}^{\mathscr{P}[k]}(t) + r^{(k)}(t) \qquad k \in \{1,\dots,T\} \tag{3.10}$$

where $r^{(k)}(t) \sim U(0,1)$ and $\{r^{(k)}(t)\}$ are independent. In this way, the particles whose index is $\{\mathscr{P}[T+k]\}_{k=1}^{T}$ is moved in a neighbourhood of the particle whose index is $\{\mathscr{P}[k]\}_{k=1}^{T}$.

Instead, when considering the scenario in Figure 3.13 (b.) the replacement rule for the particles with indices $\{\mathscr{P}[T+k]\}_{k=1}^{T}$ is the following:

$$\underline{x}^{\mathscr{P}[T+k]}(t) \leftarrow \underline{x}^{\mathscr{P}[1]}(t) + r^{(k)}(t) \qquad k \in \{1,\dots,T\} \tag{3.11}$$

where $r^{(k)}(t) \sim U(-0.5,0.5)$ and $\{r^{(k)}(t)\}$ are independent.

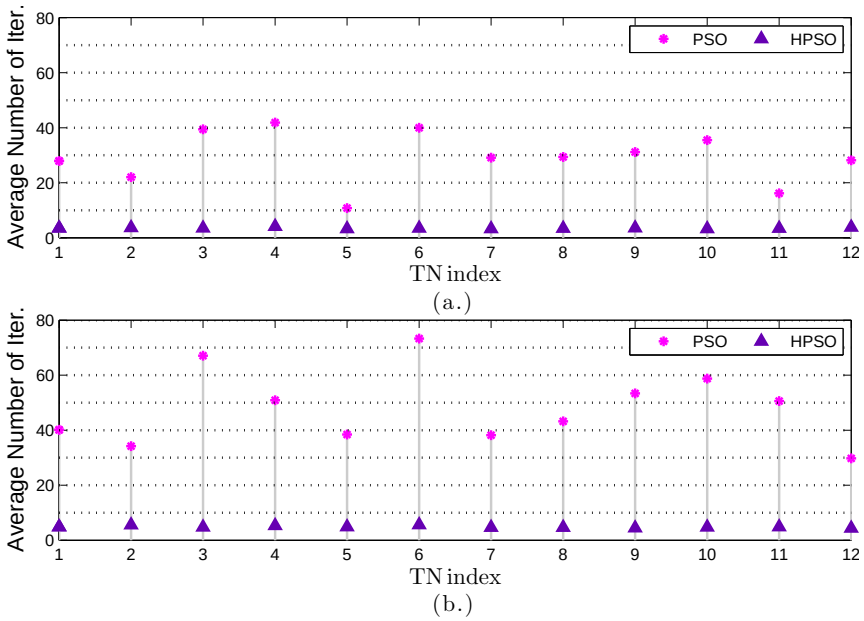


Figure 3.14: Averages of the minimum number of iterations to achieve an error tolerance of (a.) 10^{-1} m and (b.) $5 \cdot 10^{-2}$ m, for each TN in the scenario shown in Figure 3.13 (a.), when using the PSO algorithm (*magenta asterisks*), and when using the HPSO algorithm (*violet triangles*).

3.5.1 Accuracy versus Number of Iterations

We now present some simulation results concerning the minimum number of iterations required by the PSO algorithm to achieve a given error tolerance. The population size and all the other parameters are kept equal to the ones already set in Section 3.3, except, as indicated above, for the stopping criterion. The value of the inertial factor is $\omega(t) = 0$. We compare the minimum number of iterations needed to obtain the same performance when using the PSO algorithm and when using the proposed hybrid version.

Figure 3.14 refers to the scenario in Figure 3.13 (a.), where 3 ANs are used to locate 12 TNs (in sequence). Recall that in this case the relocation rule for the worst

half of the particles is (3.10). In Figure 3.14 the average number of iterations is shown as a function of the TN index in the cases with (a.) tolerance equal to 10^{-1} and (b.) tolerance equal to $5 \cdot 10^{-2}$. In both cases, the PSO and the HPSO algorithms are considered. For every TN index the HPSO algorithm converges in a far smaller number of iterations. Moreover, when the HPSO algorithm is used, the average number of iterations does not significantly change for different TNs, as it does when using the PSO algorithm.

Comparing the results in Figure 3.14 (a.) with the ones in Figure 3.14 (b.), it can be observed that, as expected, the number of iterations needed to meet the stopping criterion for the smallest tolerance is larger than when considering a higher value of the maximum tolerable RMSE.

Figure 3.15 refers to the scenario in Figure 3.13 (b.), where 4 ANs are used to locate, sequentially, 21 TNs. As stated in Section 3.5, in this scenario, the relocation rule for the worst half of the particles is (3.11).

First, we set the maximum acceptable RMSE to 10^{-1} m. The simulation results, obtained by averaging over 100 independent runs, are shown in Figure 3.15 (a.). It can be observed that the number of iterations needed to achieve the given error tolerance when using the HPSO algorithm is far smaller than the one needed when using the PSO algorithm. To be precise, the average (over all TNs) minimum number of iterations needed to achieve an RMSE equal to 10^{-1} m is 25, while this value decreases to 3 when using the proposed HPSO algorithm.

Then, we reduce the error tolerance to $5 \cdot 10^{-2}$ m: the corresponding results, are shown in Figure 3.15 (b.). Once again, the number of iterations needed to achieve the given tolerance decreases when using the HPSO algorithm instead of the PSO algorithm. In this case, the average minimum number of iterations needed to achieve the given tolerance is: 37 when using the PSO algorithm and 4 when using the proposed HPSO algorithm.

As expected, a comparison between Figure 3.15 (a.) and Figure 3.15 (b.) shows the number of iterations needed when the tolerance is smaller is larger than when considering a higher value of the maximum tolerable RMSE.

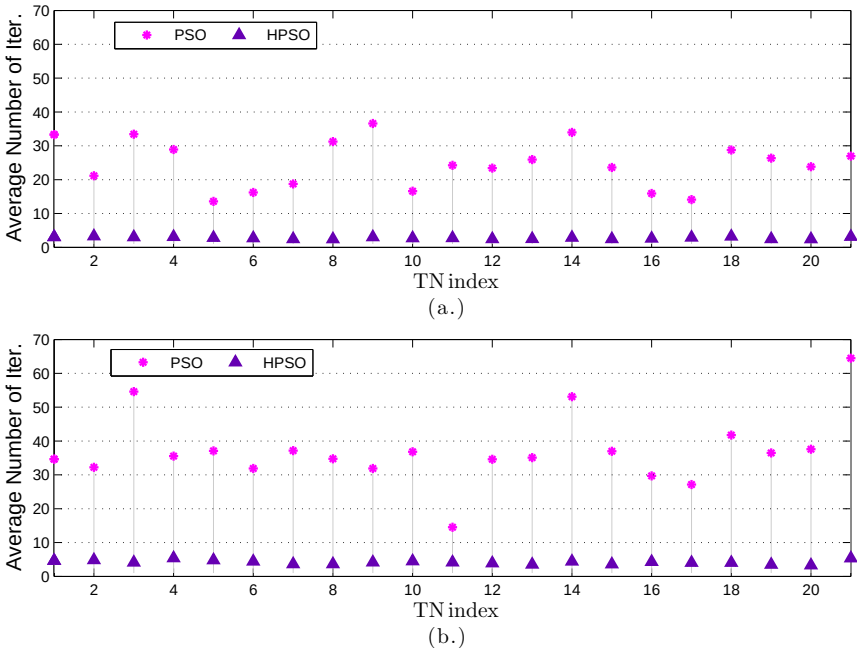


Figure 3.15: Averages of the minimum number of iterations to achieve an error tolerance of (a.) 10^{-1} m and (b.) $5 \cdot 10^{-2}$ m, for each TN in the scenario shown in Figure 3.13 (b.), when using the standard PSO algorithm (*magenta asterisks*), and when using the HPSO algorithm (*violet triangles*).

3.5.2 Impact of the Population Size

In the following, considering the same scenario shown in Figure 3.13 (a.), and setting the error tolerance to $5 \cdot 10^{-2}$ m, we evaluate, as a function of the size S of the swarm, the number of iterations needed, on average, to achieve this tolerance. Besides the already considered value $S = 40$, we also consider its half, namely $S = 20$, and its double, namely $S = 80$.

The simulation results relative to such values of S are shown in Figure 3.16. More precisely, Figure 3.16 (a.) refers to the PSO algorithm, while Figure 3.16 (b.) refers to the HPSO algorithm. In both cases, as expected, the number of iterations decreases as the population size increases.

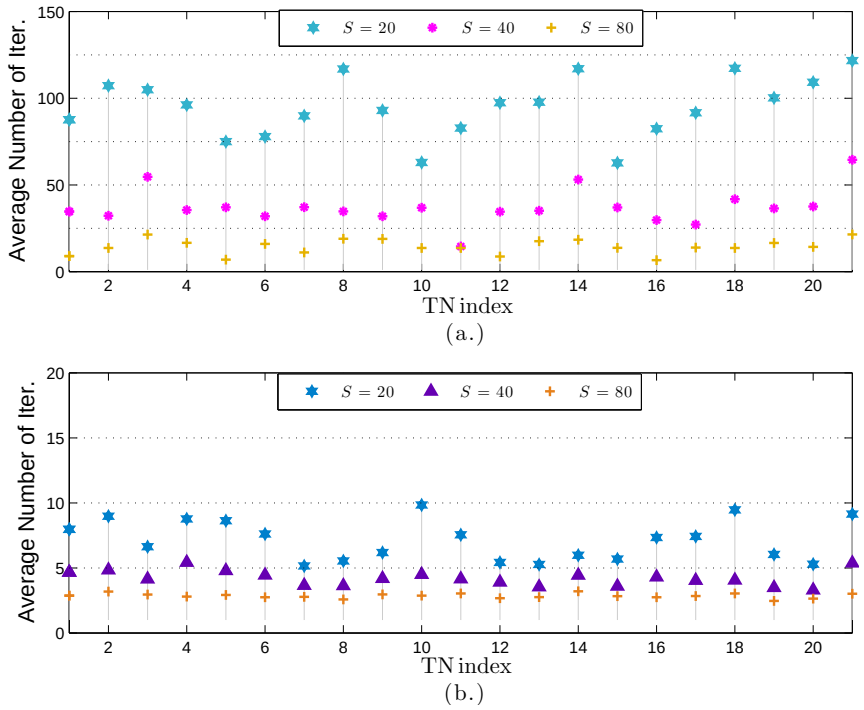


Figure 3.16: Averages of the minimum number of iterations needed to achieve an error tolerance of $5 \cdot 10^{-2}$ m are represented, for each of the 21 TNs, when using (a.) the PSO algorithm and when the population size is $S = 20$ (*cyan hexagrams*), $S = 40$ (*magenta asterisks*), and $S = 80$ (*yellow plus*) and (b.) the HPSO algorithm and when the population size is $S = 20$ (*blue hexagrams*), $S = 40$ (*violet triangles*), and $S = 80$ (*orange plus*).

Comparing the results in Figure 3.16 (a.) with those in Figure 3.16 (b.) shows that, regardless of the swarm size, the HPSO algorithm allows obtaining the same performance in a smaller number of iterations. When $S = 20$, the average number of iterations with the PSO algorithm is 95, but it reduces to 7 when using the HPSO algorithm. When $S = 40$, the average number of iterations with the PSO algorithm is 37, and it is only 4 when using the HPSO algorithm. The number of iterations with the PSO and the HPSO algorithms reduces to 14 and 3, respectively, when $S = 80$.

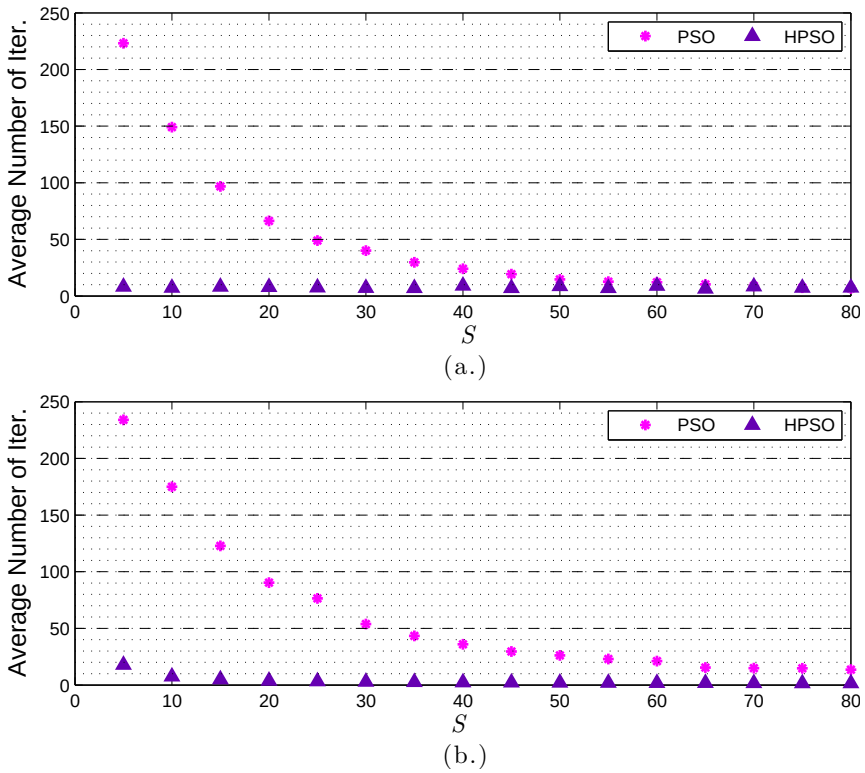


Figure 3.17: Average number of iterations, averaged over all the nodes, to achieve an error tolerance of (a.) 10^{-1} m and (b.) $5 \cdot 10^{-2}$ m, when using the standard PSO algorithm (*magenta asterisks*), and when using the HPSO algorithm (*violet triangles*).

The impact of the population size on the convergence speed is clearly highlighted in Figure 3.17, where the average number of iterations needed to achieve a maximum tolerable RMSE of (a.) 10^{-1} m and (b.) $5 \cdot 10^{-2}$ m is shown as a function of the population size S (varying from 5 to 80 individuals). Figure 3.17 shows that the proposed HPSO algorithm allows obtaining the required accuracy with a far smaller number of iterations, especially if the population size is small. The convergence of the HPSO is very fast even when the population size is small (e.g., $S = 20$ or $S = 30$), making it a very attractive choice also for real-time dynamic localization.

3.5.3 Computational Cost

We now show that, in spite of the additional step in (3.10) and/or in (3.11), the HPSO algorithm is computationally less expensive than the PSO algorithm. This gain comes mainly from the fact that the number of iterations needed to achieve a given accuracy is, as previously shown, far smaller with the HPSO algorithm.

Let us denote as N the number of TNs and as $N_{\text{iter}}^{(\text{PSO})}$ and $N_{\text{iter}}^{(\text{HPSO})}$ the numbers of iterations required by the PSO and the HPSO algorithms to converge, respectively. From the results in Figure 3.17, it can be concluded that $N_{\text{iter}}^{(\text{PSO})}$ can be roughly approximated as $1000/S$, i.e., the swarm size S needs to be very large to guarantee fast convergence. At the opposite, $N_{\text{iter}}^{(\text{HPSO})}$ is an approximately constant function of S, typically assuming values between 5 and 10 (the only exception being the case with a tolerance of $5 \cdot 10^{-2}$ m and $S = 5$, i.e., with a very small population size, where $N_{\text{iter}}^{(\text{HPSO})} = 17$).

At each iteration, the computational cost is associated with two operations carried out for each particle of the swarm at each TN: *(i)* the evaluation of the fitness function and *(ii)* the position update.

Considering the PSO algorithm, the evaluation of the fitness function defined in (3.4) for a single particle in the swarm involves $M-2$ additions and $M-1$ multiplications. Moreover, from (3.5) and (3.8) it follows that the position update of a single particle in the swarm involves 5 additions and 5 multiplications. Hence, for each particle in the swarm, at each iteration one needs to compute $M+3$ additions and $M+4$ multiplications. The computational cost over the entire swarm for the position estimate of a single TN is then obtained by multiplying the above numbers by S, obtaining $S(M+3)$ additions and $S(M+4)$ multiplications.

When considering the HPSO algorithm, the number of operations involved in the evaluation of the fitness function for the entire swarm is twice that of the PSO algorithm: as a matter of fact, at each iteration, the fitness function of each of the worst particle is evaluated twice. This leads to $1.5S(M-2)$ additions and $1.5S(M-1)$ multiplications for the evaluation of the fitness function for all particles of the swarm. The position update of each particle involves the same numbers of operations as in the PSO algorithm, with $\lfloor S/2 \rfloor$ supplementary additions, due to the relocation of half

	PSO	HPSO
Multiplications/iterations	$NS(M+4)$	$NS(3/2M+7/2)$
Additions/iterations	$NS(M+3)$	$N\lfloor S(3/2M+2)+\lfloor S/2\rfloor\rfloor$
Total number of multiplications	$NS(M+4)N_{\text{iter}}^{(\text{PSO})}$	$NS(3/2M+7/2)N_{\text{iter}}^{(\text{HPSO})}$
Total number of additions	$NS(M+3)N_{\text{iter}}^{(\text{PSO})}$	$N\lfloor S(3/2M+2)+\lfloor S/2\rfloor\rfloor N_{\text{iter}}^{(\text{HPSO})}$

Table 3.1: Numbers of multiplications and additions (per iteration and total) of the PSO and HPSO algorithms as functions of the number of TNs N, the number of particles in the swarm S, the number of ANs M, and the number of iterations ($N_{\text{iter}}^{(\text{PSO})}$ and $N_{\text{iter}}^{(\text{hPSO})}$, respectively).

of the particles given in (3.11), leading to $5S+\lfloor S/2\rfloor$ additions and $5S$ multiplications for the entire swarm at a TN. It can then be concluded that, at each iteration, the HPSO algorithm requires $S(3/2M+2)+\lfloor S/2\rfloor$ additions and $S(3/2M+7/2)$ multiplications for the position estimate of each TN.

Therefore, one obtains the computational costs in terms of numbers of multiplications and additions summarized in Table 3.1. The first two rows refer to the computational costs per iteration (obtained by multiplying the previously explained results by the number of TNs N), whereas the last two rows of Table 3.1 refer to the total costs, obtained taking into account proper numbers of iterations for the considered algorithms.

We now refer to the node configuration in Figure 3.13 (b.), where the number of TNs is $N=21$. We first consider a swarm composed of $S=30$ particles at each of the $N=21$ TNs. According to the results in Figure 3.17, in this case the numbers of iterations needed to achieve an accuracy of $5\cdot 10^{-2}$ m are $N_{\text{iter}}^{(\text{PSO})}=53$ and $N_{\text{iter}}^{(\text{HPSO})}=4$, respectively.

Then, we consider a swarm composed of $S=60$ particles at each of the $N=21$ TNs. According to the results in Figure 3.17, in this case the numbers of iterations needed to achieve an accuracy of $5\cdot 10^{-2}$ m are $N_{\text{iter}}^{(\text{PSO})}=36$ and $N_{\text{iter}}^{(\text{HPSO})}=3$, respectively.

	PSO		HPSO	
S	30	60	30	60
Total number of multiplications	267120	362880	23940	35910
Total number of additions	233730	317520	21420	32130
Simulation run time	2.08 s	2.68 s	0.20 s	0.31 s

Table 3.2: Comparison between PSO and HPSO algorithm, with $N = 21$, $M = 4$. Two possible values for the swarm size S are considered: 30 (which corresponds to $N_{\text{iter}}^{(\text{PSO})} = 53$ and $N_{\text{iter}}^{(\text{hPSO})} = 4$) and 60 (which corresponds to $N_{\text{iter}}^{(\text{PSO})} = 36$ and $N_{\text{iter}}^{(\text{hPSO})} = 3$). The numerical values predicted by the last two rows of Table 3.1 are shown. In the last row, the simulation run times are also shown.

In Table 3.2, the corresponding values of the total numbers of multiplications and additions are shown. It can be observed that, in both cases, the computational cost of the HPSO algorithm is around 10% of that of the PSO algorithm.

The last row of Table 3.2 shows the simulation execution times required by the considered algorithms to estimate the positions of the 21 TNs in the aforementioned scenario using Matlab on Mac OS 10.5.8, with a 2 GHz Intel Core 2 Duo processor. The obtained execution run times confirm the analytical prediction of Table 3.1.

Chapter 4

Experimental Characterization for UWB Distance Estimates

Exitus acta probat

– Publius Ovidius Naso

4.1 Introduction

All the localization algorithms investigated in Chapter 2 and Chapter 3 receive, as their input, a set of estimated distances. Therefore, making a single distance estimate more accurate is likely to have a positive impact on the performance of any of the considered algorithm. The goal of this chapter is, indeed, to analyze the impact of an accurate statistical model for distance estimation on the localization accuracy of practical UWB systems. Our experimental characterization is based on the use of a particular type of UWB sensors, namely the PulsON 410 Ranging and Communications Modules (RCMs) produced by Time Domain (`www.timedomain.com`).

The characterization of radio propagation channel is important to ensure satisfactory performance of a wireless communication system [10]. The existing models can

be classified into two major classes: statistical models, which rely on measured high-level data, and site-specific propagation models, based on electromagnetic wave propagation theory [63]. In this chapter, a novel statistical model for distance estimates based on experimental measurements between pairs of UWB sensors is derived.

Different kinds of experimental UWB channel models in various scenarios, have been proposed in the literature. In [53] the RSS between UWB devices is modeled on the basis of an experimental campaign performed in an indoor environment and the collected data are used to improve the accuracy of the considered localization system. Results of experimental measurements in a typical office are also shown in [7]. In this scenario, the path loss, the amplitude distribution function, and the temporal correlation between adjacent multipath components are analyzed and statistical distributions for all such parameters are given. The UWB channel in different indoor scenarios, namely, a factory hall, is investigated in [29] both in LoS and NLoS conditions. The propagation of UWB signals has been investigated also in outdoor environments, in LoS and NLoS conditions [55]. In [66] a characterization of the path loss for UWB communication in parking areas is given, while the results of a measurement campaign in a snowy environment, which may be useful in case of avalanches, are shown in [13]. Finally, a path loss model for UWB Body Area Networks (BANs), obtained by analyzing the signal propagation around the human body, is proposed in [58].

4.2 Time Domain PulsON 410 RCMs

The PulsON 410 RCM is a single-board UWB radio module intended to be integrated into users' electronic devices to enable high precision distance measurement coupled with wireless communications. The main feature of the PulsON 410 RCMs is that they provide accurate and reliable measures of inter-sensor distances at a high update rate. Two different range measurement techniques are supported, namely: Two-Way (TW) ToF ranging technique, which provides Precision Range Measurement (PRM), i.e., highly precise range measurements with high update rate; and Coarse Range Estimation (CRE) technique, according to which the range estimate is obtained from the signal strength.

Figure 4.1: A PulsON 410 RCM.

The frequency transmission of the PulsON 410 RCMs is centered at 4.3 GHz, and a bandwidth of more than 1 GHz is guaranteed, as the RF band spans from 3.1 GHz to 5.3 GHz. Each sensor is composed of a single small board (with dimensions 7.6 cm $\times$8.0 cm $\times$1.6 cm) and it needs to be connected to a power source (12 Volt). Each PulsON 410 RCM can support up to two UWB antennas [64]. A PulsON 410 RCM is shown in Figure 4.1.

Usually, the communication architecture encompasses many PulsON 410 RCMs and at least one host, connected to at least one module via USB connection. The requests are always originated from hosts and the RCMs provide confirmations for all requests. More precisely, the interface between host and RCMs consists of six request messages from host to RCM with their associated confirm messages. In addition, info messages are sent to the USB-connected host every time a UWB packet is received from other RCMs.

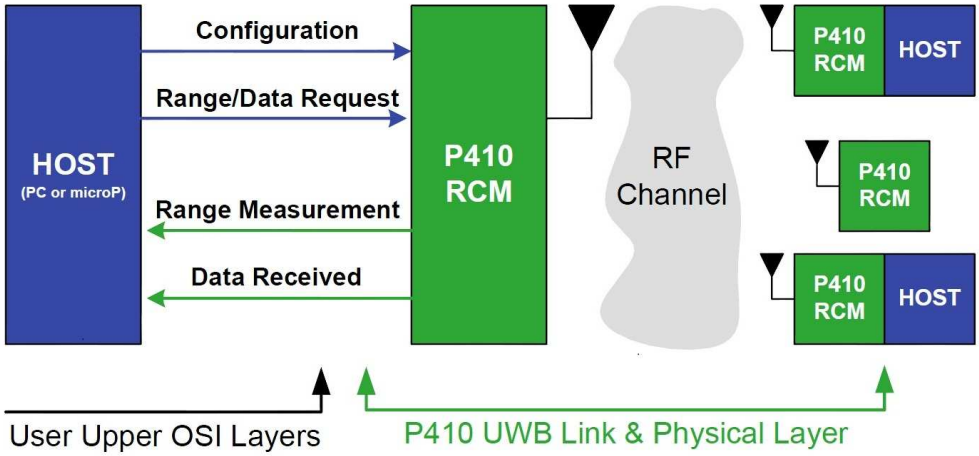


Figure 4.2: A typical PulsON 410 RCM communication architecture.

Each RCM can communicate, on the UWB channel, to all the RCMs in the considered environment, and each message exchange is associated with a message identifier. Each RCM is associated with an ID and communications between pairs of RCMs can be performed either as broadcast or by directly addressing a given ID. A typical communication architecture is shown in Figure 4.2 [64].

The TW-ToF technique is packet-based and it involves some steps. First, the requesting RCM (which is connected to a host) transmits a request packet, i.e., a long packet of pseudo-randomly encoded pulses. Once another RCM receives the packet, it transmits a response packet to the requester, which computes the precise time delay, with picoseconds accuracy, between the request packet transmission and the response packet reception.

The distance, in millimiters, between the two considered RCMs is then estimated by the receiver RCM by properly processing the time delay estimate, taking into account the antenna delay offsets. After the reception of a UWB packet, the requester RCM communicates all received data to its host, which can process it. In the following, a pair of PulsON 410 RCMs is used to derive a statistical model for the estimated distances between them.

Listing 4.1: List of C library function prototypes

```
1  int rcmIfInit(rcmIfType ifType, char *destAddr);
2  void rcmIfClose(void);
3
4  int rcmBit(int *status);
5  int rcmOpModeSet(int opMode);
6  int rcmSleepModeSet(int sleepMode);
7  int rcmConfigGet(rcmConfiguration *config);
8  int rcmConfigSet(rcmConfiguration *config);
9  int rcmStatusInfoGet(rcmMsg_GetStatusInfoConfirm *c);
10 int rcmDataSend(int mode, int size, char *data);
11 int rcmRangeTo(unsigned long destNodeId, int mode,
12     int size, char *data, rcmMsg_RangeInfo *rangeInfo,
13     rcmMsg_DataInfo *dataInfo,
14     rcmMsg_ScanInfo *scanInfo,
15     rcmMsg_FullScanInfo *fullScanInfo);
```

The PulsON 410 RCM can be programmed using C or Matlab libraries, which provide functions to support host-RCM communications via USB and serial. Here, we focus on the main C library functions, which are enumerated in Listing 4.1.

The first two functions manage the communications between the host and the connected module. More precisely

- `rcmIfInit()`: opens communication with the module. The `ifType` can be `rcmIfSerial` or `rcmIfUsb` and the `destAddr` depends on the chosen type (e.g.: `/dev/usbmodem101` for USB communications in MacOS 10.5.8).
- `rcmIfClose()`: closes the communications channel.

Once the communication channel is open, the library provides the following functions to wrap messages between the host and the RCM module

- `rcmBit()`: performs a built-in test of the module and returns the `status` of the module. If `status` equals `OK`, the module passed the built-in test.

- `rcmOpModeSet()`: sets the operation mode. It can be used to make sure that the device is configured in the proper operation mode among the available modes. Only the RCM mode is used in the presented expertiment.

- `rcmSleepModeSet()`: sets the energy saving mode. This can be used to ensure that the device is awake or to force a low consumpion mode.

- `rcmConfigGet()`: retrieves the current configuration of the module and fills the `config` structure. For instance, this function can be used to read the node ID of the device (`config.nodeId`).

- `rcmConfigSet()`: sets the configuration of the module with the `config` structure. For instance, this function can be used to set the node ID of the device.

- `rcmStatusInfoGet()`: retrieves informations about the installed hardware and firmware, and also about the current measured temperature.

- `rcmDataSend()`: can be used to send applications-specific data to the device. All RCM within range will receive the data and promiscuously send this data to their respective host (if attached). This function provides a mean to send data, over the UWB channel, to all devices and hosts in range.

- `rcmRangeTo()`: this function is used to get range informations from a TN (`destNodeId`). If `destNodeId` equals `BROADCAST_NODE_ID`, then all devices within range respond but only the first received reply is processed. Observe that, by definition, `BROADCAST_NODE_ID` is equal to `-1`. The most important return information is in the `rangeInfo` structure, which provides three types of range measurements in millimeters. Other information in `scanInfo` and `fullScanInfo` is tagged as debug-only by the official documentation. It is worth noting that together with the range request, the program can also send application-specific data which is received by all hosts in range, as in the case of `rcmDataSend`.

We have defined a modified version of the function `rcmRangeTo()` which takes the ID of the requester node as an additional input parameter. This function is denoted as `rcmRangeToEx()` and will be used in Chapter 5.

The functions previously described use data structures to pass parameters and to receive values. All such structures use platform independent data type

- `rcm_uint32_t`: unsigned 32-bit integer
- `rcm_uint16_t`: unsigned 16-bit integer
- `rcm_uint8_t`: unsigned 8-bit integer

A detailed list of the data structures that the C library provides can be found in official documentation [65]. For the sake of simplicity, we only show some of the main parameters of `rcmMsg_RangeInfo`, as they will be used in the following. Such message is sent by the local requesting RCM to its host at the end of a full UWB ranging conversation.

- `rcm_uint16_t msgId`: identifier to correlate range requests with information messages.
- `rcm_uint32_t responderId`: ID of the responding module.
- `rcm_uint8_t rangeStatus`: status/error codes for the range conversation.
- `rcm_uint8_t antennaMode`: specifies the antenna to be used during this conversation.
- `rcm_uint16_t stopwatchTime`: how long the range conversation took, in milliseconds.
- `rcm_uint32_t precisionRangeMm`: distance in millimeters between a pair of UWB modules based on a TW-ToF measurement (PRM).
- `rcm_uint32_t coarseRangeMm`: coarse distance in millimeters based on direct path signal strength (CRE). Note that this value is recalibrated to the PRM value each time a PRM of the link is measured.

- `rcm_uint32_t filteredRangeMm`: filtered distance in millimeters based on the combination of PRM and CRE values passed through a recursive optimal Kalman estimator with two state variables.

- `rcm_uint16_t vPeak`: the absolute maximum value in the leading edge window of the received waveform. This value is used to determine the CRE.

- `rcm_uint32_t timestamp`: milliseconds since the module boot at the time of the range conversation.

In the experimental results shown in this dissertation, the localization algorithms are performed on the basis of the distances `rcm_uint32_t precisionRangeMm`. We remark that `vPeak` is a quality metrics which accompanies any range message. The `vPeak` is a measure of the maximum absolute value of the amplitude measured just after the leading edge offset. The leading edge offset is a parameter associated with the direct sequential scan of the pulse waveform. More precisely, the leading edge offset is the index in the scan where the system determines the ToA of the pulse. The parameter `vPeak` is an improved form of RSS and it rarely suffers from multipath fading [65].

4.3 Distance Estimate Model

In this section, on the basis of an extensive experimental measurement campaign, we derive a statistical model for the distance estimate, in terms of its average value and its standard deviation. We assume that the following equation holds

$$\hat{r} = r + v(r) \tag{4.1}$$

where r and $\hat{r}$ represent the true and estimated distances, respectively, and $v(r)$ is the error. We model the error as a random variable and we assume that its statistical distribution depends on the true distance between the two considered sensors. The validity of this assumption will be shown later in this section, by analyzing the collected experimental data.

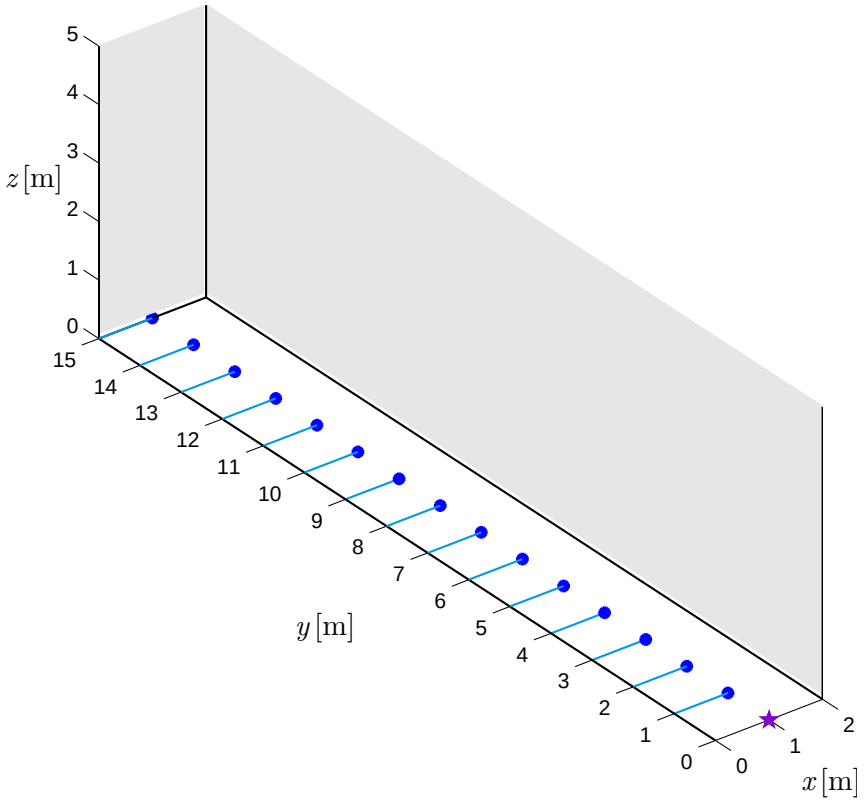


Figure 4.3: The scenario considered in the experiments: the position of the sensor connected to the host is shown (*purple star*) and the 15 positions of the second sensor are shown (*blue dots*).

In order to derive a statistical model for the distance estimate, we perform distance measurements between two RCMs. We connect one of them (namely, the requester) to a host, while the second one (namely, the receiver) is not connected to any host. We take distance estimates every meter, considering distances from 1 m to 15 m. For every considered distance, we take 1000 (independent) distance estimates. All the data are collected in a corridor whose width is 2 m and whose height is 5 m. The considered scenario is shown in Figure 4.3, where the star represents the position of the requester and the dots represent the considered receiver positions.

Let us denote as r_k, $k \in \{1,\ldots,15\}$, the true distances between the two sensors, namely, $r_k = k$ m and as $\hat{r}_k^{(j)}$ the distance estimate relative to r_k in the $j-$th iteration, $j \in \{1,\ldots,1000\}$. Assuming that the distance estimates are affected by additive noise, for each distance r_k and for each iteration j, one can then write

$$\hat{r}_k^{(j)} = r_k + \hat{v}_k^{(j)} \tag{4.2}$$

where $\hat{v}_k^{(j)}$ is the distance error realization in the $j-$th iteration. For a given value of r_k, it is reasonable to assume that $\{\hat{v}_k^{(j)}\}_{j=1}^{1000}$ are independent and identically distributed realizations, as they are relative to independent distance measurements.

Let us define as $\{\bar{v}_k\}_{k=1}^{15}$ and $\{\sigma_k\}_{k=1}^{15}$ the average value and the standard deviation of the distance estimates based on the data collected in the 1000 iterations

$$\bar{v}_k \triangleq \frac{1}{1000}\sum_{j=1}^{1000}\hat{v}_k^{(j)} \tag{4.3}$$

$$\sigma_k \triangleq \sqrt{\frac{1}{999}\sum_{j=1}^{1000}\left(\hat{v}_k^{(j)} - \bar{v}_k\right)^2}. \tag{4.4}$$

The values of $\{\bar{v}_k\}_{k=1}^{15}$ and $\{\sigma_k\}_{k=1}^{15}$, corresponding to the distances $\{r_k\}_{k=1}^{15}$, are shown in Table 4.1. It can be observed that for distances r_k from 1 m to 10 m the average error $\bar{v}_k$ is negative, meaning that the estimated distances $\hat{r}_k^{(j)}$ are, on average, shorter than the true ones. At the opposite, for distances larger than 10 m the average error $\bar{v}_k$ assumes positive values, which means that the true distances are, on average, overestimated.

More generally, from Table 4.1 it can be noticed that the average error $\bar{v}_k$ tends to be an increasing function of the distance r_k between the two sensors. In order to derive a simple statistical model for the values $\{\bar{v}_k\}_{k=1}^{15}$, we consider the following linear approximation

$$\bar{v}_k^{(\mathrm{LS})} \simeq a r_k + b. \tag{4.5}$$

The coefficients a and b that appear in (4.5) can be found by solving the linear system

$$\underline{\underline{A}}\,\underline{t} = \underline{v} \tag{4.6}$$

r_k [m]	$\bar{v}_k$ [mm]	σ_k [mm]
1	-151.7	29.8
2	-71.8	18.1
3	-89.1	20.1
4	-135.0	12.2
5	-76.9	22.2
6	-40.2	31.8
7	-19.0	8.1
8	-43.3	20.5
9	-30.9	24.1
10	-12.4	25.2
11	4.4	23.2
12	90.5	65.4
13	127.9	66.1
14	88.1	23.5
15	27.7	47.5

Table 4.1: The values of $\bar{v}_k$ (second column) and σ_k (third column) as functions of the distances r_k.

where $\underline{\underline{A}}$ is a 15×2 matrix with the $k-$th element of the first column given by r_k and the elements of the second column all equal to 1, the $k-$th element of $\underline{v}$ is $\bar{v}_k$, expressed in meters, and $\underline{t} = [a,b]^T$ is the solution vector. Since A is a rectangular matrix, the system (4.6) can be solved through the LS technique. The LS solution of the system (4.6) is

$$\underline{t} \simeq \begin{pmatrix} 0.016 \\ -0.150 \end{pmatrix} \tag{4.7}$$

so that the average error $\bar{v}_k$, can be modeled as

$$\bar{v}_k^{(\mathrm{LS})} = 0.016\, r_k - 0.150 \tag{4.8}$$

where it is assumed that $\{\bar{v}_k^{(\mathrm{LS})}\}_{k=1}^{15}$ and $\{r_k\}_{k=1}^{15}$ are expressed in meters.

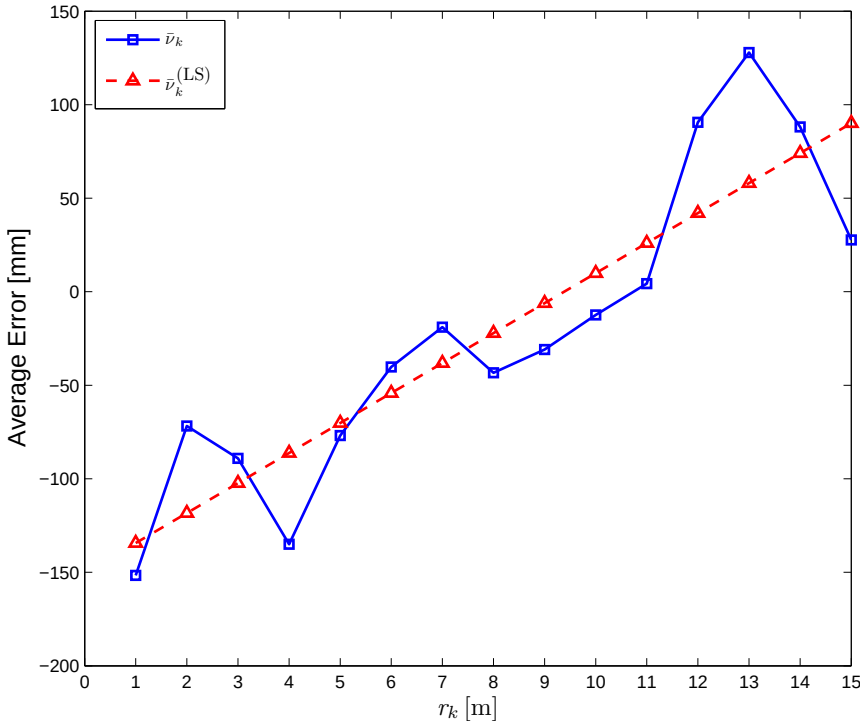


Figure 4.4: The values, in millimiters, of error $\bar{v}_k$ (*blue squares*) and of its linear approximation $\bar{v}_k^{(\mathrm{LS})}$ (*red triangles*) are shown as a function of the true distance r_k between the two sensors.

A graphical representation of the average error $\bar{v}_k$ and of its approximation $\bar{v}_k^{(\mathrm{LS})}$ obtained in (4.8) is shown in Figure 4.4. The blue squares represent the values of $\bar{v}_k$, expressed in millimiters, while the red triangles represent the values of the linear approximation $\bar{v}_k^{(\mathrm{LS})}$.

From Figure 4.4 it can be observed that the linear approximation is accurate. As a matter of fact, the absolute value of the difference between $\bar{v}_k$ and $\bar{v}_k^{(\mathrm{LS})}$ is, on average, 30 mm and it is always smaller than 70 mm.

Generalizing (4.8), one can write

$$\bar{v}^{(\mathrm{LS})}(r) = 0.016\,r - 0.150. \tag{4.9}$$

Given that the linear approximation of $\overline{v}_k^{(\mathrm{LS})}$ as a function of r_k fits well the experimental data $\overline{v}_k$, the generic expression of the distance error $v(r)$ in (4.1) can be replaced with $\overline{v}^{(\mathrm{LS})}(r)$ given by (4.9) and the estimated distance $\hat{r}$ can be modeled as

$$\hat{r} \simeq r + \overline{v}^{(\mathrm{LS})}(r) = 1.016r - 0.150. \tag{4.10}$$

From Table 4.1 it can be observed that the standard deviation is always smaller than 70 mm and it is, on average, 29 mm. Such small values of the standard deviation guarantee accurate distance estimates. As done for the average value, we now consider the following linear approximation for the stardard deviation $\{\sigma_k\}_{k=1}^{15}$

$$\sigma_k^{(\mathrm{LS})} \simeq mr_k + q. \tag{4.11}$$

The coefficients m and q in (4.11) can be found by solving the system

$$\underline{\underline{A}}\,\underline{v} = \underline{\sigma} \tag{4.12}$$

where $\underline{\underline{A}}$ is the same matrix as in (4.6), the $k-$th element of $\underline{\sigma}$ is σ_k, expressed in meters, and $\underline{v} = [m,q]^T$ is the solution vector. The system (4.12) can be solved once again through the LS technique. The LS solution of the system (4.12) is

$$\underline{v} \simeq \begin{pmatrix} 0.002 \\ 0.012 \end{pmatrix} \tag{4.13}$$

so that the standard deviation σ_k, can be modeled as

$$\sigma_k^{(\mathrm{LS})} = 0.002\, r_k + 0.012 \tag{4.14}$$

where it is assumed that $\sigma_k^{(\mathrm{LS})}$ and r_k are expressed in meters.

The standard deviation σ_k and its approximation $\sigma_k^{(\mathrm{LS})}$ are shown in Figure 4.5. The cyan triangles represent the values of σ_k, expressed in millimiters, while the magenta squares represent the values of the linear approximation $\sigma_k^{(\mathrm{LS})}$. From Figure 4.5 it can be observed that the linear approximation is accurate. As a matter of fact, the absolute value of the difference between σ_k and $\sigma_k^{(\mathrm{LS})}$ is, on average, 11 mm and it is always smaller than 28 mm.

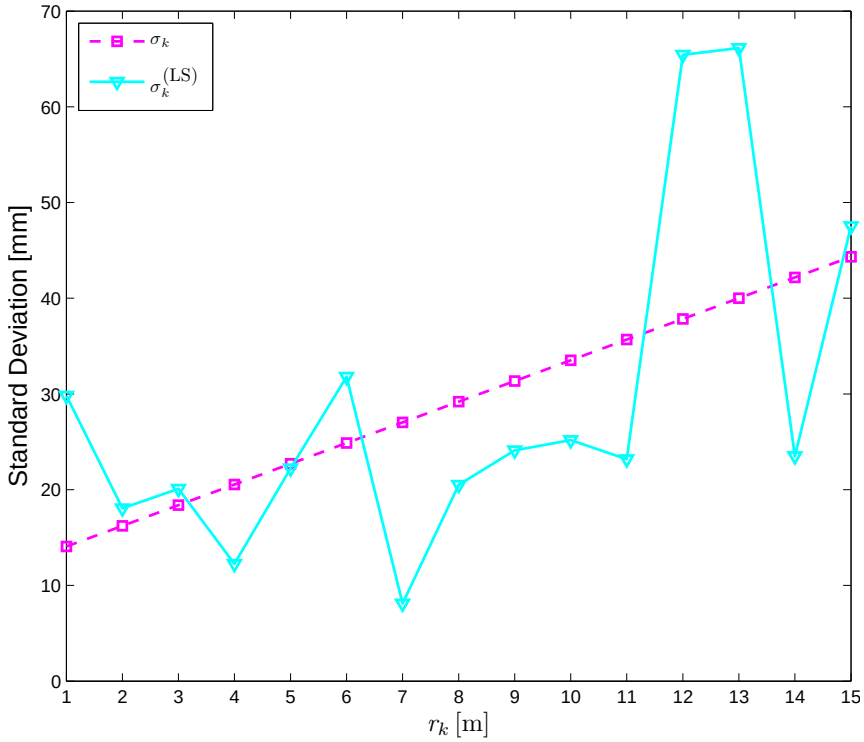


Figure 4.5: The values, in millimiters, of σ_k (*cyan triangles*) and its linear approximation $\sigma_k^{(\mathrm{LS})}$ (*magenta squares*) are shown as a function of the true distance r_k between the two sensors.

Generalizing (4.14), one can write

$$\sigma^{(\mathrm{LS})} = 0.002\,r + 0.012. \tag{4.15}$$

To summarize, we can conclude that the distance error $v(r)$ introduced in (4.1) can be modeled as a random variable, whose average value $\overline{v}$ and standard deviation σ depend of the true distance r between the two considered sensors, according to

$$\overline{v}(r) \simeq v^{(\mathrm{LS})} = 0.016r - 0.150 \tag{4.16}$$

$$\sigma(r) \simeq \sigma^{(\mathrm{LS})} = 0.002r + 0.012 \tag{4.17}$$

where all the quantities are expressed in meters.

4.4 Localization

In this section, we describe the localization setup which is used to validate the statistical model derived in Section 4.3 and to obtain the results shown in Section 4.5. More precisely, we consider four PulsON 410 RCMs in a bidimensional scenario. Assuming to know the positions of three of them (namely, the ANs), we aim at finding the (unknown) position of the remaining one (namely, the TN). We denote the coordinates of the ANs as

$$\underline{s}_i \triangleq [x_i, y_i] \qquad i \in \{1,2,3\} \tag{4.18}$$

while the coordinates of the TN are indicated as

$$\underline{u} \triangleq [x, y]. \tag{4.19}$$

In order to gather experimental results, we connect the PulsON 410 RCM which corresponds to the TN to a host. Proper Matlab libraries are used for the communication between the TN and its host. First, the TN collects the range estimates from the three ANs by directly interrogating, in sequence, the IDs corresponding to each of the three ANs. Once the TN receives the range estimates from all the ANs, recalling that the coordinates of the ANs are known, it can compute its position by means of proper localization algorithms.

In the following, two localization scenarios are described and, for each of them, two localization algorithms are considered, namely: the Circumference Intersection (CI) algorithm, described in Subsection 4.4.2, and the TSML-ToA algorithm, described in Subsection 1.3.2.

4.4.1 Indoor Localization Scenarios

The experimental results which will be shown in Section 4.5 are relative to two different scenarios. The main difference between the two scenarios is that in the first case the TN lies inside the convex hull of the three ANs, while in the latter the TN lies outside the convex hull of the three ANs.

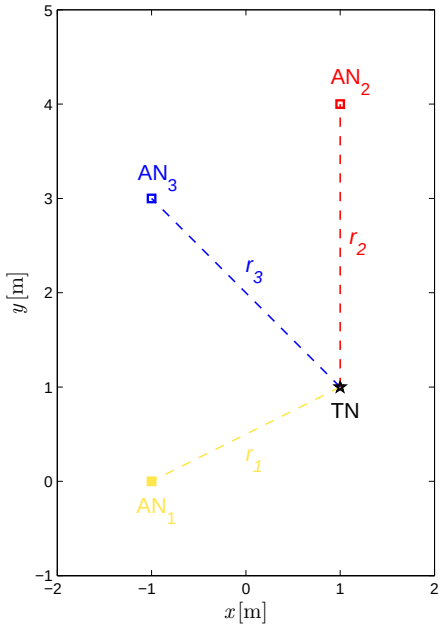


Figure 4.6: The position of the TN (*black star*) is shown, together with the positions of three ANs (*squares*). The distances $\{r_i\}_{i=1}^{3}$ are also shown.

First, we consider a scenario where the coordinates of the ANs, expressed in meters, are

$$\underline{s}_1 = [-1,0] \qquad \underline{s}_2 = [1,4] \qquad \underline{s}_3 = [-1,3] \tag{4.20}$$

while the position of the TN is identified by

$$\underline{u} = [1,1]. \tag{4.21}$$

From (4.20) and (4.21), the values of the distances $\{r_i\}_{i=1}^{3}$, expressed in meters, are

$$r_1 = \sqrt{5} \qquad r_2 = 3 \qquad r_3 = 2\sqrt{2}. \tag{4.22}$$

In Figure 4.6 the considered scenario is shown: coloured squares represent the positions of the three ANs, while the position of the TN is identified by the black star. The (true) distances $\{r_i\}_{i=1}^{3}$ between each AN and the TN are also shown.

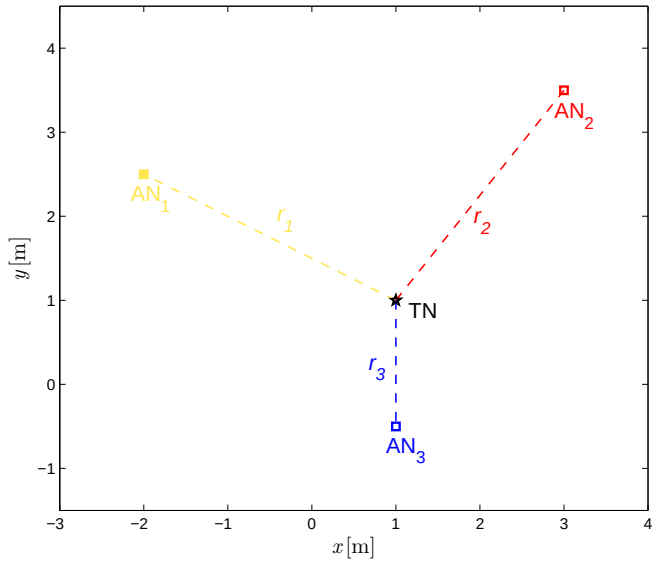


Figure 4.7: The position of the TN (*black star*) is shown, together with the positions of three ANs (*squares*). The distances $\{r_i\}_{i=1}^{3}$ are also shown.

Then, we consider a scenario where the coordinates of the ANs are

$$\underline{s}_1 = [-2, 2.5] \qquad \underline{s}_2 = [3, 3.5] \qquad \underline{s}_3 = [1, -0.5] \tag{4.23}$$

and the position of the TN is the same as in (4.21), so that the values of the distances $\{r_i\}_{i=1}^{3}$, expressed in meters, are

$$r_1 = \frac{3}{2}\sqrt{5} \qquad r_2 = \frac{\sqrt{41}}{2} \qquad r_3 = 1.5. \tag{4.24}$$

In Figure 4.7 the considered scenario is shown: coloured squares represent the positions of the three ANs, while the position of the TN is identified by the black star. The (true) distances $\{r_i\}_{i=1}^{3}$ between each AN and the TN are also shown.

In order to statistically characterize the accuracy of the position estimates for both the scenarios, we consider 1000 location estimations. In the following, we denote as $\hat{r}_i^{(j)}$ the estimated distance between $\{\mathrm{AN_i}\}_{i=1}^{3}$, and the TN in the $j-$th iteration and as $\underline{\hat{u}}^{(j)} \triangleq [\hat{x}^{(j)}, \hat{y}^{(j)}]$ the estimated TN coordinates in the $j-$th iteration, $j \in \{1, \ldots, 1000\}$.

4.4.2 Circumference Intersection (CI) Algorithm

In this subsection, a simple localization algorithm, denoted as Circumference Intersection (CI), is presented. A similar approach is introduced in [49]. We choose to start from this algorithm because it is one of the most intuitive and it can be extended to more elaborate algorithms.

In order to better explain the CI algorithm, let us make a few geometric considerations on the considered problem. The knowledge of the coordinates of the ANs and of the true distances $\{r_i\}_{i=1}^3$ would allow obtaining the true position of the TN. As a matter of fact, the TN lies on each of the circumferences $\{\mathscr{C}_i\}_{i=1}^3$, centered in $\{\underline{s}_i\}_{i=1}^3$ and with radii $\{r_i\}_{i=1}^3$ identified by the following equations

$$\begin{cases} (x-x_1)^2+(y-y_1)^2=r_1^2 \\ (x-x_2)^2+(y-y_2)^2=r_2^2 \\ (x-x_3)^2+(y-y_3)^2=r_3^2. \end{cases} \tag{4.25}$$

Hence, the TN coordinates $\underline{u}=[x,y]$ could be found as the (unique) point where the three circumferences $\{\mathscr{C}_i\}_{i=1}^3$ intersect.

Since the true distances $\{r_i\}_{i=1}^3$ are unknown, one can only rely on their estimates $\{\hat{r}_i^{(j)}\}_{i=1}^3$, $j\in\{1\ldots,1000\}$. For this reason, at each step $j\in\{1,\ldots,1000\}$, equations (4.25) are replaced by

$$\begin{cases} (x-x_1)^2+(y-y_1)^2=\left[\hat{r}_1^{(j)}\right]^2 \\ (x-x_2)^2+(y-y_2)^2=\left[\hat{r}_2^{(j)}\right]^2 \\ (x-x_3)^2+(y-y_3)^2=\left[\hat{r}_3^{(j)}\right]^2. \end{cases} \tag{4.26}$$

Equations in (4.26) refer to the three circumferences, denoted as $\{\hat{\mathscr{C}}_i^{(j)}\}_{i=1}^3$, centered in $\{\underline{s}_i\}_{i=1}^3$ with radii $\{\hat{r}_i^{(j)}\}_{i=1}^3$. While the three circumferences $\{\mathscr{C}_i\}_{i=1}^3$ intersect in a unique point (which corresponds to the true TN position), due to the errors that affect the range estimates $\{\hat{r}_i^{(j)}\}_{i=1}^3$, the three circumferences $\{\hat{\mathscr{C}}_i^{(j)}\}_{i=1}^3$ may not intersect in a unique point.

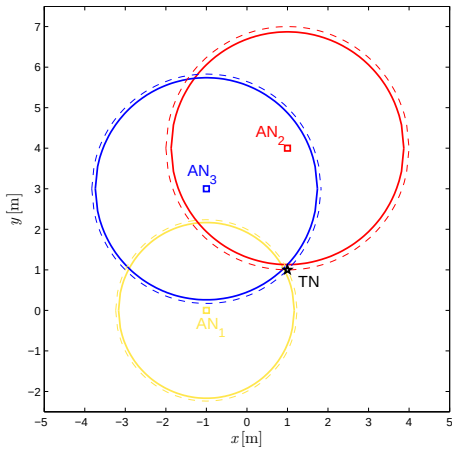


Figure 4.8: The three ANs (*squares*) and the TN (*black star*) are shown. The three circumferences $\{\hat{\mathscr{C}}_i^{(1)}\}_{i=1}^3$ obtained in the first iteration are shown (*solid lines*) together with the circumferences $\{\mathscr{C}_i\}_{i=1}^3$ (*dashed lines*).

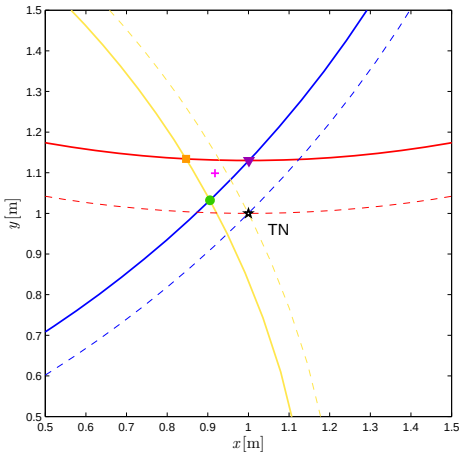


Figure 4.9: A zoom of Figure 4.8 in a neighbourhood of the TN is shown. The intersections between pairs of circumferences are emphasized, and the obtained position estimate (*magenta plus*) is shown.

In Figure 4.8, the circumferences $\{\mathscr{C}_i\}_{i=1}^3$ defined in (4.25) and referred to the scenario in Figure 4.6 are shown (dashed lines). As an illustrative example, the circumferences $\{\hat{\mathscr{C}}_i^{(1)}\}_{i=1}^3$ relative to the distance estimates $\{\hat{r}_i^{(1)}\}_{i=1}^3$, (i.e., the distance estimates obtained in the first iteration) are also shown (solid lines). The TN position is denoted as a black star.

Even if, from Figure 4.8, the circumferences (4.25) with radii equal to the true distances $\{r_i\}_{i=1}^3$ may seem very similar to the corresponding circumferences (4.26) with radii equal to $\{\hat{r}_i^{(1)}\}_{i=1}^3$ (i.e., corresponding to $j=1$), Figure 4.9, which represents a zoom of Figure 4.8 in a neighbourhood of the TN, shows that there is indeed a relevant difference. As a matter of fact, as expected, Figure 4.9 shows that: *(i)* the position of the TN coincides with the (unique) intersection between the three circumferences defined in (4.25) and *(ii)* the three circumferences defined in (4.26) do not intersect in a single point.

In order to find the TN position estimate in the $j-$th iteration, which is denoted as $\underline{\hat{u}}^{(j)} = [\hat{x}^{(j)}, \hat{y}^{(j)}]$, it is therefore necessary to consider a proper localization approach: instead of looking for the intersection of the three circumferences (4.26), we intersect pairs of them (namely, $\binom{3}{2} = 3$ pairs).

More precisely, for each iteration $j \in \{1, \ldots, 1000\}$, let us define the following three sets (each of which contains two points), obtained by intersecting the three different pairs of circumferences

$$I_1 = \{\underline{p}_{12}, \underline{q}_{12}\} \triangleq \hat{\mathscr{C}}_1^{(j)} \cap \hat{\mathscr{C}}_2^{(j)} \tag{4.27}$$

$$I_2 = \{\underline{p}_{13}, \underline{q}_{13}\} \triangleq \hat{\mathscr{C}}_1^{(j)} \cap \hat{\mathscr{C}}_3^{(j)} \tag{4.28}$$

$$I_3 = \{\underline{p}_{23}, \underline{q}_{23}\} \triangleq \hat{\mathscr{C}}_2^{(j)} \cap \hat{\mathscr{C}}_3^{(j)}. \tag{4.29}$$

We then choose a point from each of the three sets, namely $\underline{p}_1 \in I_1$, $\underline{p}_2 \in I_2$, and $\underline{p}_3 \in I_3$, so that the three selected points are the nearest ones to each other. More precisely, since such points belong to circumferences intersections, it can be shown that they can be chosen as

$$||\underline{p}_1 - \underline{p}_2|| = \min_{\underline{p} \in I_1, \underline{q} \in I_2} ||\underline{p} - \underline{q}|| \tag{4.30}$$

$$||\underline{p}_1 - \underline{p}_3|| = \min_{\underline{q} \in I_3} ||\underline{p}_1 - \underline{q}||. \tag{4.31}$$

With reference to the configuration in Figure 4.8, such three points are shown in Figure 4.9. In particular, in Figure 4.9 the orange square corresponds to one of the two intersections between $\hat{\mathscr{C}}_1^{(1)}$ and $\hat{\mathscr{C}}_2^{(1)}$, the green asterisk corresponds to one of the two intersections between $\hat{\mathscr{C}}_1^{(1)}$ and $\hat{\mathscr{C}}_3^{(1)}$, and the violet triangle corresponds to one of the two intersections between $\hat{\mathscr{C}}_2^{(1)}$ and $\hat{\mathscr{C}}_3^{(1)}$.

Finally, the TN position estimate (magenta plus in Figure 4.9) is found as the baricenter of these three points.

We remark that the intersection between two circumferences could also be an empty set. In this case, the TN position estimate is found based on the remaining intersections. A modified and improved version of this algorithm is explain in detail in Chapter 5 for a different scenario.

4.5 Application of the Statistical Model

In this section, the statistical model for the distance estimates derived in Section 4.3 is applied to practical localization problems, namely the ones introduced in Subsection 4.4.1, in order to analyze the impact of the statistical characterization of the range estimation error on the accuracy of the obtained position estimate.

Observe that, from the statistical model for the distance error (4.10), the true distance r between a pair of RCMs can be expressed as a function of the estimated distance $\hat{r}$ according to

$$r \simeq \frac{\hat{r}+0.150}{1.016}. \tag{4.32}$$

It is then of interest to investigate what happens if, instead of applying the localization strategies with the distances $\{\hat{r}_i^{(j)}\}_{i=1}^3$ (namely, the ones estimated from the RCMs), we apply them to the following distances, obtained by correcting $\{\hat{r}_i^{(j)}\}_{i=1}^3$ with formula (4.32)

$$\check{r}_i^{(j)} \triangleq \frac{\hat{r}_i^{(j)}+0.150}{1.016} \qquad i \in \{1,2,3\}. \tag{4.33}$$

We show how the statistical model for the distance estimates derived in Section 4.3 is effective in improving the performance of the considered localization algorithms, namely the CI localization algorithm, explained in the previous section, and the TSML-ToA algorithm.

These algorithms take as input the estimated distances between the ANs and the TN and, therefore, the localization accuracy strongly depends on the quality of such distance estimates.

In the following, for each scenario we first apply the considered localization strategies with the distances $\{\hat{r}_i^{(j)}\}_{i=1}^3$ estimated from the RCMs. For each configuration of the nodes, 1000 distance estimates between each AN and the TN are taken and they are then used in the aforementioned localization algorithms to obtain 1000 position estimates of the TN. Let us denote such position estimates as

$$\underline{\hat{u}}^{(j)} \triangleq [\hat{x}^{(j)}, \hat{y}^{(j)}] \qquad j \in \{1,\ldots,1000\}.$$

Then, we do the same with the distance corrections $\{\check{r}_i^{(j)}\}_{i=1}^3$ defined in (4.33), obtained by applying to the same distance estimates $\{\hat{r}_i^{(j)}\}_{i=1}^3$ the statistical model (4.10), thus obtaining 1000 TN position estimates, denoted as

$$\underline{\check{u}}^{(j)} \triangleq [\check{x}^{(j)}, \check{y}^{(j)}] \qquad j \in \{1, \ldots, 1000\}.$$

We then compare the accuracy of the position estimates obtained with the distances $\{\hat{r}_i^{(j)}\}_{i=1}^3$ with the accuracy of the position estimates obtained with their corrections $\{\check{r}_i^{(j)}\}_{i=1}^3$. The comparison is based on the average distance and the maximum distance between the TN position estimates and the true TN position.

In particular, denote as $\hat{d}^{(j)}$ the distance between the true TN position and its estimate in the $j-$th iteration without taking into account the statistical model of the range error and we denoted as $\check{d}^{(j)}$ the distance between the true TN position and its estimate in the $j-$th iteration when using the proposed statistical model. Therefore, one can write

$$\begin{aligned} \hat{d}^{(j)} &\triangleq ||\underline{u} - \underline{\hat{u}}^{(j)}|| \\ \check{d}^{(j)} &\triangleq ||\underline{u} - \underline{\check{u}}^{(j)}||. \end{aligned} \tag{4.34}$$

Then, the average values (over the 1000 iterations) of the distances between true and estimated TN positions defined in (4.34) can be defined as

$$\begin{aligned} \hat{d}_{\text{avg}} &\triangleq \frac{1}{1000} \sum_{j=1}^{1000} \hat{d}^{(j)} \\ \check{d}_{\text{avg}} &\triangleq \frac{1}{1000} \sum_{j=1}^{1000} \check{d}^{(j)}. \end{aligned} \tag{4.35}$$

Finally, we also define the maximum values of the distances between true and estimated TN positions defined in (4.34), namely

$$\begin{aligned} \hat{d}_{\text{max}} &\triangleq \max_{j \in \{1,\ldots,1000\}} \hat{d}^{(j)} \\ \check{d}_{\text{max}} &\triangleq \max_{j \in \{1,\ldots,1000\}} \check{d}^{(j)}. \end{aligned} \tag{4.36}$$

The following experimental results are relative to the two scenarios introduced in Figure 4.6 and Figure 4.7 in Section 4.3.

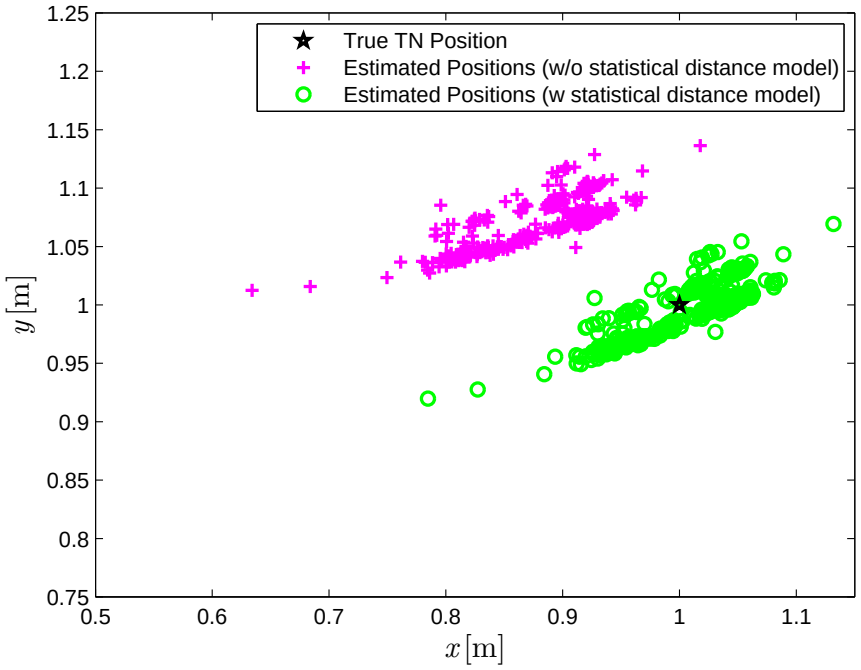


Figure 4.10: The 1000 position estimates obtained with the CI localization algorithm (*magenta plus*) and the 1000 position estimates obtained when taking into account the statistical model for range estimates (*green circles*) are shown. The TN (*black star*) is also shown.

4.5.1 First Scenario

The results in this subsection refer to the scenario shown in Figure 4.6, hence the one in which the TN lies outside the convex hull of the three ANs. We first consider the CI algorithm. In Figure 4.10, we directly compare the true TN position (black star) with the position estimates obtained in 1000 measurements with (green circles) and without (magenta pluses) the correction (4.33). From Figure 4.10, it is apparent that the proposed statistical model allows improving the performance accuracy of the CI localization algorithm. As a matter of fact, the TN position estimates obtained with the distance corrections (4.33) are closer to the true TN position.

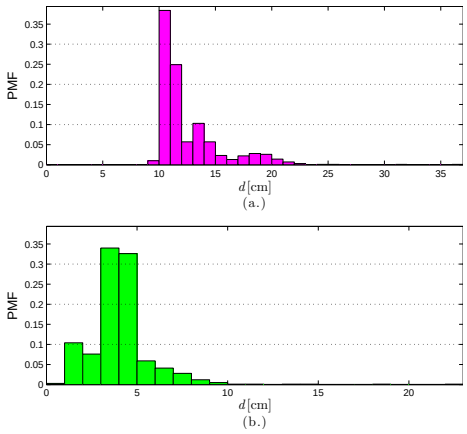


Figure 4.11: PMFs of the distance errors in the scenario in Figure 4.10 (a.) without and (b.) with the statistical model.

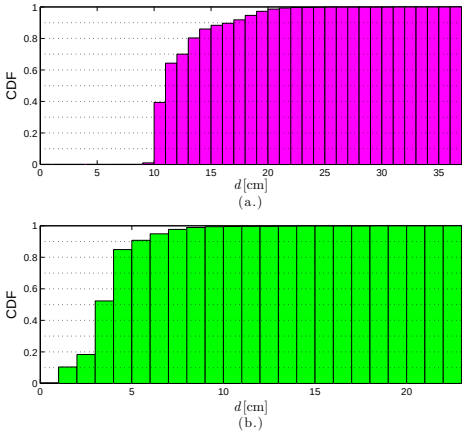


Figure 4.12: CDFs of the distance errors in the scenario in Figure 4.10 (a.) without and (b.) with the statistical model.

In Figure 4.11, the Probability Mass Functions (PMFs) of the distances between the true TN position and its estimates are shown. More precisely, Figure 4.11 (a.) concerns the position estimates obtained without the statistical model for the distance estimates. In this case, the distances between the true TN position and its estimates are almost always greater than 10 cm. Figure 4.11 (b.), instead, refers to the position estimates obtained with the statistical model. In this case, the distances between the true TN position and its estimates are never greater than 10 cm.

In Figure 4.12, the Cumulative Distribution Functions (CDFs) of the distances between the true TN position and its estimates are shown. Figure 4.12 (a.) is relative to the position estimates obtained without the statistical model and Figure 4.12 (b.) is relative to the position estimates obtained by applying the correction (4.33). Figure 4.12 shows that, when applying the proposed correction, the distance error is smaller than 7 cm more than 95% of the times. At the opposite, if the statistical model for the distance estimates is not used, the distance error is never smaller than 8 cm. Moreover, in order to guarantee that the distance error is smaller than a given threshold 95% of the times, one needs to increase the distance threshold to 19 cm.

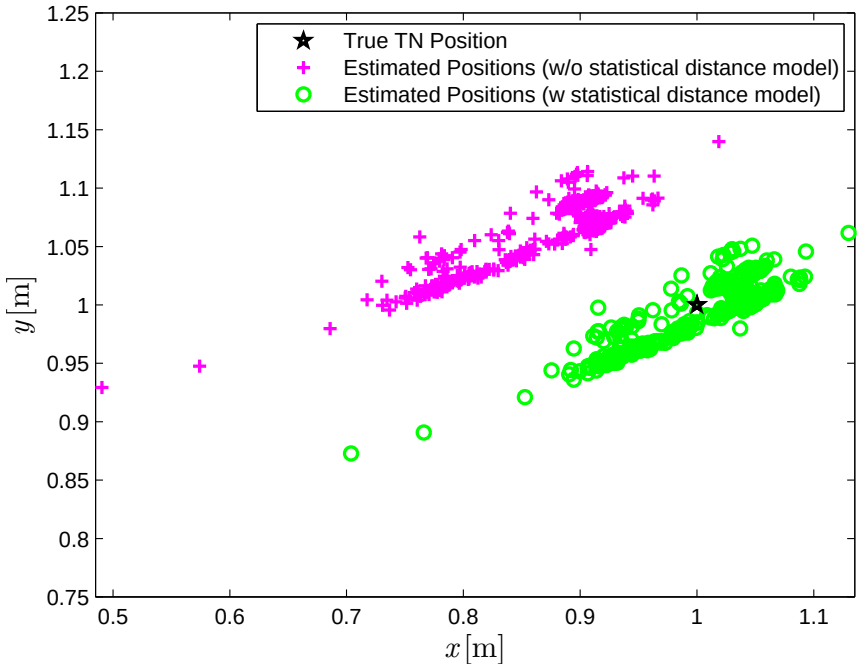


Figure 4.13: The 1000 position estimates obtained with the TSML-ToA localization algorithm (*magenta plus*) and the 1000 position estimates obtained when taking into account the statistical model for range estimates (*green circles*) are shown. The TN (*black star*) is also shown.

We now consider the TSML-ToA algorithm in the scenario shown in Figure 4.6 and we apply it to the same range estimates $\{\hat{r}_i^{(j)}\}_{i=1}^3$ and $\{\check{r}_i^{(j)}\}_{i=1}^3$ used with the CI algorithm. In Figure 4.13, the results obtained with the TSML-ToA localization algorithm are shown. The position estimates obtained in 1000 independent runs of the TSML-ToA algorithm with (green circles) and without (magenta pluses) the distance correction (4.33) are shown. The true TN position (black star) is also shown for comparison purposes. The results are analogous to those obtained in Figure 4.10 with the CI algorithm. As a matter of fact, from Figure 4.13 it can be observed that the use of the proposed statistical model improves the performance accuracy of the TSML-ToA localization algorithm as well.

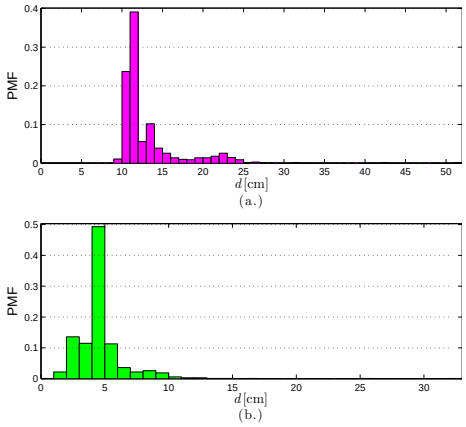


Figure 4.14: PMFs of the distance in the scenario in Figure 4.13 (a.) without and (b.) with the statistical model.

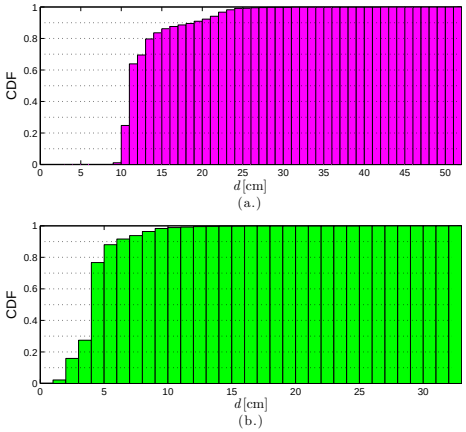


Figure 4.15: CDFs of the distance errors in the scenario in Figure 4.13 (a.) without and (b.) with the statistical model.

In Figure 4.14 and Figure 4.15, the PMFs and the CDFs of the distances between the true TN position and its estimates (obtained according to the TSML-ToA algorithm) are shown.

In particular, Figure 4.14 (a.) shows the PMF of the distance between the true TN position and its estimates obtained without taking into account the statistical model for the distance estimates. At the opposite, Figure 4.14 (b.) shows the PMF of the distance between the true TN position and its estimates obtained when considering the distance correction (4.33). These PMFs are both similar to those shown in Figure 4.11 obtained with the CI algorithm.

Figure 4.15 (a.) shows the CDF relative to the position estimate error obtained without the statistical model for the distance estimates, while Figure 4.15 (b.) is relative to the position estimates obtained using the correction (4.33). The two CDFs are similar to those in Figure 4.12 and the distance error when applying the proposed correction turns out to be smaller than 9 cm more than 95% of the times. At the opposite, if the model for the distance estimates is ignored, the distance error is never smaller than 9 cm and it is smaller than 10 cm only 1% of the times.

Scenario 1	CI Algorithm	TSML-ToA Algorithm
$\hat{d}_{\text{avg}}$ [m]	0.126	0.131
$\check{d}_{\text{avg}}$ [m]	0.041	0.046
$\hat{d}_{\text{max}}$ [m]	0.366	0.514
$\check{d}_{\text{max}}$ [m]	0.229	0.322

Table 4.2: The average and the maximum distances between the true TN position and its estimates with ($\hat{d}_{\text{avg}}$, $\hat{d}_{\text{max}}$) and without ($\check{d}_{\text{avg}}$, $\check{d}_{\text{max}}$) the use of the statistical model. The CI algorithm and the TSML-ToA algorithms are considered.

Let us now consider Table 4.2, where the values of the average distance and the maximum distance between the true TN position and its estimates are shown. The second column of Table 4.2 refers to the CI algorithm while the third column of Table 4.2 refers to the TSML-ToA algorithm.

From the second column of Table 4.2, it can be observed that the use of the values of the distances $\{\check{r}_i\}_{i=1}^{3}$, obtained with the correction (4.33), instead of values of the distances $\{\hat{r}_i\}_{i=1}^{3}$ directly estimated by the nodes, allows reducing the average range error. As a matter of fact, the value of $\check{d}_{\text{avg}}$ is 8.5 cm lower than $\hat{d}_{\text{avg}}$ (corresponding to a reduction of 67%). Similarly, the maximum range error $\check{d}_{\text{max}}$, obtained using $\{\check{r}_i\}_{i=1}^{3}$, is 14 cm shorter (corresponding to a reduction of 37%) than the maximum range error $\hat{d}_{\text{max}}$ obtained using $\{\hat{r}_i\}_{i=1}^{3}$.

The third column of Table 4.2 shows that, as already observed, the distance correction (4.33) also improves the performance of the TSML-ToA algorithm. As a matter of fact, it can be observed that also in this case the value of $\check{d}_{\text{avg}}$ is 8.5 cm smaller (corresponding to a reduction of 64%) than $\hat{d}_{\text{avg}}$, while the value of $\check{d}_{\text{max}}$ is nearly 19 cm (corresponding to a reduction of 37%) smaller than $\hat{d}_{\text{max}}$.

The performances of the two algorithms are then similar in terms of the average distance between the true TN position and its estimates, while the values of $\hat{d}_{\text{max}}$ and $\check{d}_{\text{max}}$ are larger when using the TSML-ToA algorithm with respect to those obtained with the CI algorithm.

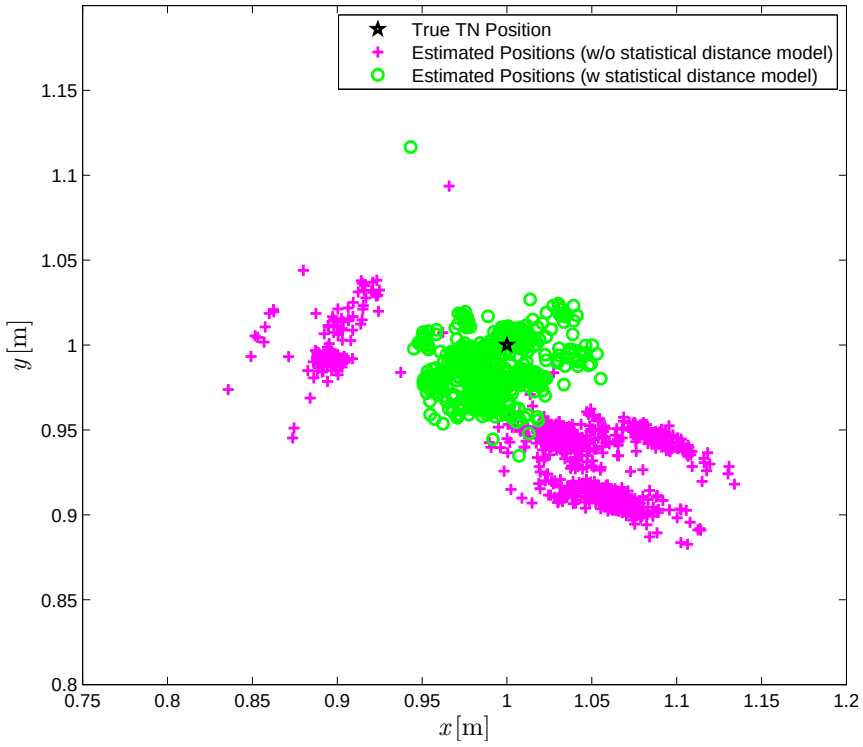
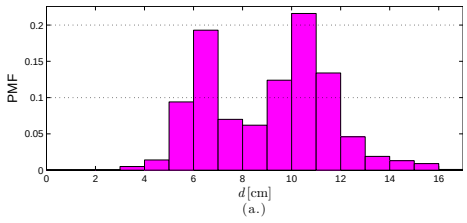


Figure 4.16: The 1000 position estimates obtained with the CI localization algorithm (*magenta plus*) and the 1000 position estimates obtained when taking into account the statistical model for range estimates (*green circles*) are shown. The TN (*black star*) is also shown.

4.5.2 Second Scenario

The results presented in this subsection refer to the scenario shown in Figure 4.7 where the TN lies inside the convex hull of the three ANs. We start by considering the CI algorithm. In Figure 4.16, we directly compare the true TN position (black star) with the position estimates obtained over 1000 measurements with (green circles) or without (magenta pluses) the correction (4.33). From Figure 4.16, it is apparent that the proposed statistical model allows improving the performance accuracy of the CI localization algorithm.



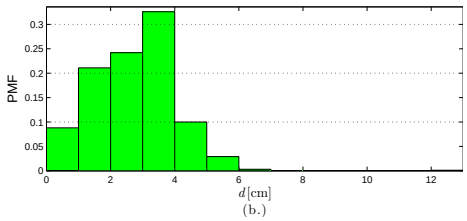
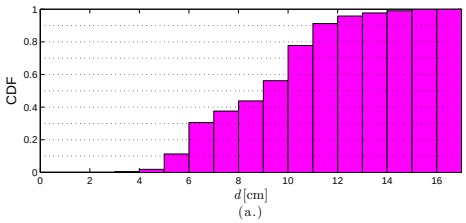


Figure 4.17: PMFs of the distance errors in the scenario in Figure 4.16 (a.) without and (b.) with the statistical model.



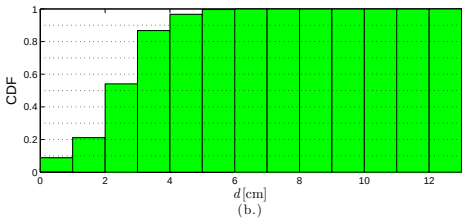


Figure 4.18: CDFs of the distance errors in the scenario in Figure 4.16 (a.) without and (b.) with the statistical model.

In Figure 4.17 and Figure 4.18, the PMFs and the CDFs of the distances between the true TN position and its estimates are shown.

Figure 4.17 (a.) shows the PMF of the distance between the true TN position and its estimates when the statistical model for the distance estimates is not used, while Figure 4.17 (b.) concerns the PMF obtained when taking into account the statistical model defined in (4.33). Observe that these distributions are different from those in Figure 4.11, which are relative to the scenario in Figure 4.6. More precisely, the values of the distances between the true TN position and its estimates in Figure 4.17 are smaller than those in Figure 4.11.

Figure 4.18 (a.) shows the CDF relative to the position estimates obtained without the statistical model for the distance estimates, while Figure 4.18 (b.) is relative to the position estimates obtained applying the proposed correction (4.33). From Figure 4.18, it can be observed that, when applying the proposed correction, the distance error is smaller than 5 cm more than 95% of the times. At the opposite, if the statistical model for the distance estimates is not used, the distance error is smaller than 5 cm only 1% of the times.

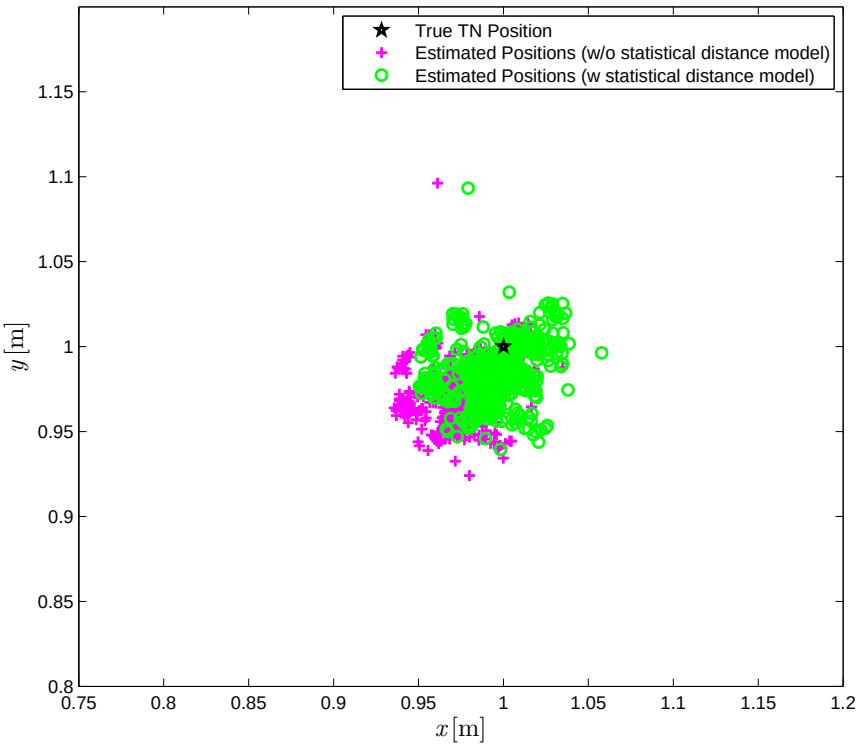


Figure 4.19: The 1000 position estimates obtained with the TSML-ToA localization algorithm (*magenta plus*) and the 1000 position estimates obtained when taking into account the statistical model for range estimates (*green circles*) are shown. The TN (*black star*) is also shown.

In Figure 4.19, the performance of the TSML-ToA localization algorithm in the scenario in Figure 4.7 is investigated. The true TN position (black star) is compared with the position estimates obtained over 1000 iterations with (green circles) or without (magenta pluses) the correction (4.33). From Figure 4.19 it can be observed that, once again, the use of the proposed statistical model improves the performance accuracy of the TSML-ToA localization algorithm. However, in this case the improvement is limited if compared to the results in Figure 4.16 obtained in the same scenario with the CI algorithm.

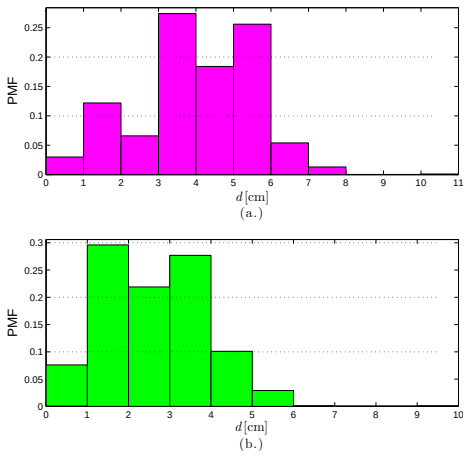


Figure 4.20: PMFs of the distance errors in the scenario in Figure 4.19 (a.) without and (b.) with the statistical model.

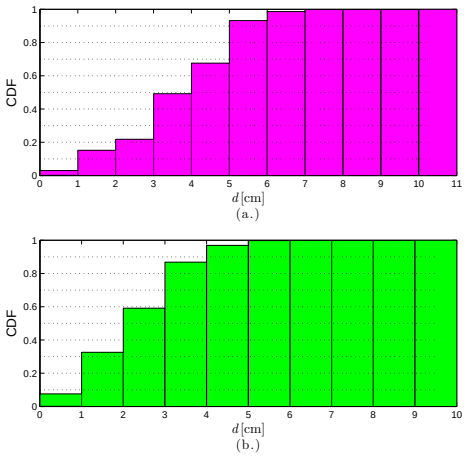


Figure 4.21: CDFs of the distance errors in the scenario in Figure 4.19 (a.) without and (b.) with the statistical model.

In Figure 4.20, the PMFs of the distances between the true TN position and its estimates are shown. More precisely, Figure 4.20 (a.) concerns the results obtained without the statistical model in (4.33), while Figure 4.20 (b.) is relative to the results obtained with the statistical model for distance estimates. As observed when considering the CI algorithm, the values of the distances between the true TN position and its estimates in Figure 4.20 are smaller than those in Figure 4.14 obtained with the same algorithm in the scenario in Figure 4.6. This suggests that a configuration of the nodes where the TN lies inside the convex hull of the three ANs is more favorable for both localization approaches.

In Figure 4.21, the CDFs of the distances between the true TN position and its estimates (obtained according to the TSML-ToA algorithm) are shown. More precisely, Figure 4.21 (a.) is relative to the position estimates obtained without the statistical model for the distance estimates, while Figure 4.21 (b.) is relative to the position estimates obtained using the correction (4.33). Unlike Figure 4.18, the two CDFs are very similar and, in both cases, the distance error turns out to be smaller than 6 cm more than 95% of the times.

Scenario 2	CI Algorithm	TSML-ToA Algorithm
$\hat{d}_{\text{avg}}$ [m]	0.090	0.040
$\check{d}_{\text{avg}}$ [m]	0.028	0.026
$\hat{d}_{\text{max}}$ [m]	0.166	0.104
$\check{d}_{\text{max}}$ [m]	0.129	0.096

Table 4.3: The average and the maximum distances between the true TN position and its estimates with ($\hat{d}_{\text{avg}}$, $\hat{d}_{\text{max}}$) and without ($\check{d}_{\text{avg}}$, $\check{d}_{\text{max}}$) the use of the statistical model. The CI algorithm and the TSML-ToA algorithm are considered.

In Table 4.3, the values of the average distance and the maximum distance between the true TN position and its estimates (relative to the second scenario) are shown. The second column of Table 4.3 refers to the CI algorithm while the third column refers to the TSML-ToA algorithm.

From the second column of Table 4.3, it can be observed that using the values of the distances $\{\check{r}_i\}_{i=1}^{3}$, obtained with the correction (4.33), instead of the values of the distances $\{\hat{r}_i\}_{i=1}^{3}$ directly estimated by the nodes, reduces the average range error. As a matter of fact, the value of $\check{d}_{\text{avg}}$ is 6 cm lower than $\hat{d}_{\text{avg}}$ (corresponding to a reduction of 68%). Moreover, the maximum range error $\check{d}_{\text{max}}$, obtained using $\{\check{r}_i\}_{i=1}^{3}$ is 4 cm shorter (corresponding to a reduction of 22%) than the maximum range error $\hat{d}_{\text{max}}$ obtained using $\{\hat{r}_i\}_{i=1}^{3}$.

From the third column of Table 4.3, instead, it can be observed that the value of $\check{d}_{\text{avg}}$ is only 1.4 cm smaller (corresponding to a reduction of 35%) than $\hat{d}_{\text{avg}}$, while the value of $\check{d}_{\text{max}}$ is only 8 mm (corresponding to a reduction of 8%) shorter than $\hat{d}_{\text{max}}$.

Chapter 5

A Localization Algorithm for Realistic Scenarios

Difficile est tenere quae acceperis
nisi exerceas

– Gaius Plinius Secundus

5.1 Introduction

In this chapter, we propose a realistic localization algorithm which tackles the issues of real-world indoor localization and we also show some experimental results obtained with the PulsON 410 RCMs in an indoor scenario with obstacles between ANs and a TN. The algorithm uses multiple distance measurements to obtain TN position estimates also in the presence of NLoS phenomena.

In our experimental setup, three or four RCMs are used as ANs and they are all connected to the same host, namely a Raspberry Pi, on which two variants of the proposed localization algorithm are implemented. Another RCM is then used as TN. Even though the performance of the proposed localization algorithm is shown in the

presence of a static TN, the proposed algorithm can also be used in dynamic real-time localization scenarios. The ANs and the TN are positioned on the same plane.

We first consider a scenario with three ANs and then a scenario with four ANs. In each scenario, N_c (independent) range estimates from each AN are taken, thus leading to N_c position estimates of the considered TN. Denoting as $\underline{u} = [x,y]$ the true position of the TN and as $\underline{\hat{u}}^{(j)} = [\hat{x}^{(j)},\hat{y}^{(j)}]$ its estimate in the $j-$th iteration, $j \in \{1,\ldots,N_c\}$, the distance (namely, the error) between the true TN position and its estimate in the $j-$th iteration can be expressed as

$$d^{(j)} \triangleq \sqrt{(x-\hat{x}^{(j)})^2 + (y-\hat{y}^{(j)})^2}. \tag{5.1}$$

In order to evaluate the performance of the proposed localization algorithm, the average distance d_{avg} over a sufficiently large number N_c of iterations is defined

$$d_{\text{avg}} \triangleq \frac{1}{N_c}\sum_{j=1}^{N_c} d^{(j)}. \tag{5.2}$$

In the following sections, some variants of the proposed localization algorithm are explained and their performances, in terms of d_{avg}, are shown.

5.2 Algorithm with Three ANs

In this section, three RCMs are considered as ANs. We remark that three is the minimum number of ANs that allows finding the position of a TN in a bidimensional scenario. We assume that the coordinates of the ANs in the considered coordinate system are

$$\underline{s}_0 = [0,L] \qquad \underline{s}_1 = [-V,0] \qquad \underline{s}_2 = [V,0] \tag{5.3}$$

where L and V are setting parameters. We also assume the presence of an obstacle, namely a large metal infrastructure, which introduces NLoS phenomena.

The considered scenario is shown in Figure 5.1, where the black rectangle represents the metal obstacle, the three ANs are denoted with coloured stars, and the values of the parameters L and V in (5.3) are set to 2 m and 0.5 m, respectively (as in the experimental scenario described in Section 5.3).

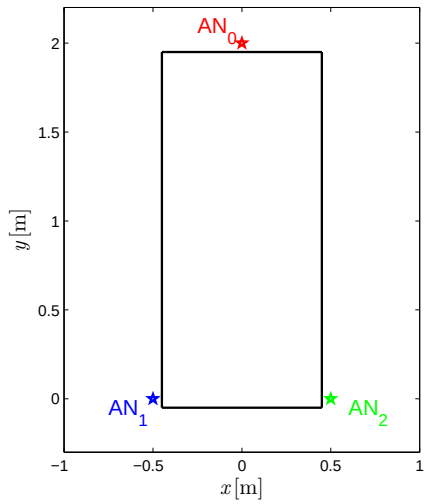


Figure 5.1: The ANs (*coloured stars*) and the metal obstacle in the first scenario (*black rectangle*) are shown.

The TN position $\underline{u}$ can be found as the intersection of the circumferences $\{\mathscr{C}_i\}_{i=0}^{2}$

$$\begin{cases} x^2+(y-L)^2=r_0^2 \\ (x+V)^2+y^2=r_1^2 \\ (x-V)^2+y^2=r_2^2 \end{cases} \tag{5.4}$$

centered in $\{\underline{s}_i\}_{i=0}^{2}$ with radii $\{r_i\}_{i=0}^{2}$ equal to the true distances between the TN and the $i-$th AN. Assuming that such distances are unknown, the localization must be based on their estimates. Let us denote as $\{\hat{r}_i^{(j)}\}_{i=0}^{2}$ the estimates of $\{r_i\}_{i=0}^{2}$ in the $j-$th iteration. Hence, the position estimate $\underline{\hat{u}}^{(j)}$ in the $j-$th iteration has to be found on the basis of the set of circumferences $\{\mathscr{C}_i^{(j)}\}_{i=0}^{2}$

$$\begin{cases} \hat{\mathscr{C}}_0^{(j)} : x^2+(y-L)^2=[\hat{r}_0^{(j)}]^2 \\ \hat{\mathscr{C}}_1^{(j)} : (x+V)^2+y^2=[\hat{r}_1^{(j)}]^2 \\ \hat{\mathscr{C}}_2^{(j)} : (x-V)^2+y^2=[\hat{r}_2^{(j)}]^2 \end{cases} \tag{5.5}$$

where the true distances $\{r_i\}_{i=0}^{2}$ in (5.4) are replaced by their estimates $\{\hat{r}_i^{(j)}\}_{i=0}^{2}$.

Two variants of the proposed localization algorithm are described in the remainder of this section. The first one only relies on the estimated distances between the ANs and the TN and it is an improved version of the CI algorithm described in Subsection 4.4.2. The second one can be considered as an improved version of the first one, as it also uses the values of `vPeak`[1] provided by the RCMs, in order to decide if a distance measurement is reliable or not. Both algorithms can be divided into two steps: the first one corresponds to the acquisition of the distance estimates between each AN and the TN; the second one performs the localization on the basis of the data collected in the first step.

The algorithms are explained in detail in the two following subsections. For the sake of simplicity, while explaining the algorithms we omit the superscript (j), used to denote the $j-$th iteration, and we simplify the notation writing $[\star]$ instead of $[\hat{\star}]$ even if the considered quantities are estimates. We remark that this notation does not introduce ambiguity because the algorithm only makes use of estimated quantities.

5.2.1 A Distance-based Localization Algorithm

In this subsection, a distance-based localization algorithm is explained. A pseudocode of the algorithm, which is implemented on the host (namely, the Raspberry Pi connected to the three RCMs) is shown in Listing 5.1. In the following, we refer to this algorithm as Algorithm 1.

In the first part of the algorithm, the distance estimates between each AN and the TN are acquired. First, the nodes AN_1 and AN_2 are put in the sleep mode and the node AN_0 performs a range request in broadcast, so that each RCM within range can reply (line 5 of Listing 5.1). As soon as a node replies, the obtained distance estimate `rangeInfo.precisionRangeMm` is saved in r_0 and the ID of the replying node `rangeInfo.responderId` is passed to AN_1 and AN_2, which are waken up sequentially to directly interrogate the replying node (lines 7 and 10 of Listing 5.1). Observe that, since a metal obstacle is placed among the RCMs, AN_1 and/or AN_2 may be in NLoS with the RCM which replied to AN_0. Due to propagation errors, it

[1] A definition of `vPeak` is given in Section 4.2.

Listing 5.1: Pseudocode of Algorithm 1 for three ANs.

```
1  procedure algorithm 1
2
3  for cycle ∈ {1,...,N_c}
4      status_1 ← 0,  status_2 ← 0
5      rcmRangeToEx(AN_0, BROADCAST_NODE_ID)
6      r_0 ← rangeInfo.precisionRangeMm
7      if (rcmRangeToEx(AN_1, rangeInfo.responderId)≠ERR)
8          status_1 ← 1
9          r_1 ← rangeInfo.precisionRangeMm
10     if (rcmRangeToEx(AN_2, rangeInfo.responderId)≠ERR)
11         status_2 ← 1
12         r_2 ← rangeInfo.precisionRangeMm
13
14     if (status_1 = 1 ∧ status_2 = 1)
15         findTargetPosition
16     else if (status_1 = 1 ∧ status_2 = 0)
17         findTargetPositionNoS2
18     else if (status_1 = 0 ∧ status_2 = 1)
19         findTargetPositionNoS1
20     else if (status_1 = 0 ∧ status_2 = 0)
21         u ← [∞,∞]
```

is then possible that the enquired TN does not reply to the range request from one or both of them. In this case, the values of $status_1$ and/or $status_2$ are kept equal to 0, otherwise they are set to 1 and the distance estimates are saved in r_1 and/or r_2, respectively.

In the second part of the program, on the basis of the distance estimates previously obtained, localization is performed. Depending on the number of available range estimates, three different localization strategies are considered.

Before analyzing in detail these localization strategies whose pseudocodes are shown in Listings 5.5, 5.6, and 5.7, we describe three frequently used functions.

Listing 5.2: Pseudocode of function `findNearer`.

```
function I = findNearer(A,B)
input: A = {p_0, p_1} coordinates of two points
       B = {q_0, q_1} coordinates of two points
output: I = {p, q} coordinates of the two nearest points
        from A and B

choose p ∈ A, q ∈ B: ||p − q|| = min_{i,j∈{0,1}} ||p_i − q_j||
I ← {p, q}
```

Listing 5.3: Pseudocode of function `maxAbscissa`.

```
function u = maxAbscissa(I)
input: I = {p_0 = [x_0, y_0], p_1 = [x_1, y_1]} coordinates of two points
output: u chosen point

if (x_0 < x_1)
    u ← p_1
else
    u ← p_0
```

The function `findNearer` defined in Listing 5.2 takes as inputs two sets (say A and B), each of which contains the coordinates of two points, and returns the two nearest points from the two sets, namely $\underline{p} \in A$ and $\underline{q} \in B$ which satisfy

$$||\underline{p} - \underline{q}|| = min_{\underline{a} \in A, \underline{b} \in B} ||\underline{a} - \underline{b}||. \tag{5.6}$$

The function `maxAbscissa` defined in Listing 5.3 takes as input a set containing the coordinates of two points and it returns the coordinate of the point with the greatest abscissa.

Similarly, the function `minAbscissa` defined in Listing 5.4 takes as input a set containing the coordinates of two points and it returns the coordinate of the point with the smallest abscissa.

Listing 5.4: Pseudocode of function `minAbscissa`.

```
1  function u = minAbscissa(I)
2  input: I = {p_0 = [x_0,y_0], p_1 = [x_1,y_1]} coordinates of two points
3  output: u chosen point
4
5  if (x_0 < x_1)
6      u ← p_0
7  else
8      u ← p_1
```

We can now describe the position estimate phase in Listing 5.1. We remark that every time the TN position $\underline{u}$ cannot be estimated, in the pseudocode we denote it as $\underline{u} = [\infty, \infty]$. Four cases are identified, on the basis of the number of available distance estimates.

Case 1. If $status_1 = 1$ and $status_2 = 1$ (line 14 of Listing 5.1) the localization algorithm is described in the procedure `findTargetPosition` in Listing 5.5. In this case, the three distance measurements $\{r_i\}_{i=0}^{2}$ are available and the final position estimate can be obtained from the intersection of pairs of circumfereces. More precisely, let us define the following two sets (lines 3 and 4 of Listing 5.5)

$$I = \mathscr{C}_0 \cap \mathscr{C}_1 \qquad J = \mathscr{C}_0 \cap \mathscr{C}_2. \tag{5.7}$$

Each set can either contain the coordinates of two points[2] if the considered circumferences intersect or it can be empty if the circumferences do not intersect.

If both pairs of circumferences intersect (line 5 of Listing 5.5), we choose the two nearest points from I and J, according to Listing 5.2, and the final position estimate is found as the baricenter of these two points (line 7 of Listing 5.5).

If $\mathscr{C}_0$ and $\mathscr{C}_2$ intersect but I is empty (line 8 of Listing 5.5), we look for the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$. The reason for doing so is that $\mathscr{C}_0$ and $\mathscr{C}_1$ may not intersect just because of small errors on the range estimates r_0 and r_1. In order to find

[2]The two points would coincide in case of tangent circumferences.

Listing 5.5: Pseudocode of procedure `findTargetPosition`.

```
1  procedure findTargetPosition
2
3  I ← {p_0, p_1} = C_0 ∩ C_1
4  J ← {q_0, q_1} = C_0 ∩ C_2
5  if(I ≠ ∅ ∧ J ≠ ∅)
6      {p, q} ← findNearer(I, J)
7      u ← 1/2 (p + q)
8  else if(I = ∅ ∧ J ≠ ∅)
9      A ← C_0 ∩ R_01, B ← C_1 ∩ R_01, I = {p_0, p_1} ← findNearer(A, B)
10     if (||p_0 − p_1|| ≤ d_Th)
11         {p, q} ← findNearer(I, J), u ← 1/2 (p + q)
12     else
13         u ← maxAbscissa(J)
14 else if(I ≠ ∅ ∧ J = ∅)
15     A ← C_0 ∩ R_02, B ← C_2 ∩ R_02, J = {q_0, q_1} ← findNearer(A, B)
16     if (||q_0 − q_1|| ≤ d_Th)
17         {p, q} ← findNearer(I, J), u ← 1/2 (p + q)
18     else
19         u ← minAbscissa(I)
20 else if(I = ∅ ∧ J = ∅)
21     A ← C_0 ∩ R_01, B ← C_1 ∩ R_01, I = {p_0, p_1} ← findNearer(A, B)
22     A ← C_0 ∩ R_02, B ← C_2 ∩ R_02, J = {q_0, q_1} ← findNearer(A, B)
23     d_1 ← ||p_0 − p_1||, d_2 ← ||q_0 − q_1||
24     if (d_1 ≤ d_Th ∧ d_2 ≤ d_Th)
25         {p, q} ← findNearer(I, J), u ← 1/2 (p + q)
26     else if (d_1 ≤ d_Th ∧ d_2 > d_Th)
27         u ← 1/2 (p_0 + p_1)
28     else if (d_1 > d_Th ∧ d_2 ≤ d_Th)
29         u ← 1/2 (q_0 + q_1)
30     else
31         u ← [∞, ∞]
```

the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$, we need to intersect both these circumferences with the line $\mathscr{R}_{01}$ which goes through their centers, whose equation is

$$\mathscr{R}_{01} : y = \frac{L}{V}x + L. \tag{5.8}$$

Each of the two intersections $\mathscr{C}_0 \cap \mathscr{R}_{01}$ and $\mathscr{C}_1 \cap \mathscr{R}_{01}$ returns two points and the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$ can be found according to the function defined in Listing 5.2. If the distance between the points is below a given distance threshold d_{Th} (line 10 of Listing 5.5), then I is redefined as the set containing these two points and the localization is performed as in the first case. Otherwise, it is reasonable to assume that the distance estimate r_1 is far inaccurate and this may be likely due to the fact that the TN and AN_1 are in NLoS. For this reason, in order to be coherent with the fact that AN_1 is obscured by the obstacle, the final position estimate $\underline{u}$ is defined as the point in J with the greatest abscissa (line 13 of Listing 5.5).

Similar considerations hold if $\mathscr{C}_0$ and $\mathscr{C}_1$ intersect but J is empty (line 14 of Listing 5.5). In this case, the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_2$ can be found by intersecting the two circumferences with the line $\mathscr{R}_{02}$ which goes through their centers

$$\mathscr{R}_{02} : y = -\frac{L}{V}x + L. \tag{5.9}$$

If the distance between these two points is below the distance threshold d_{Th} (line 16 of Listing 5.5), then J is redefined as the set containing the two points and the localization is performed as in the first case. Otherwise, since the points in I are symmetric with respect to the line $\mathscr{R}_{01}$ defined in (5.8), the final position estimate $\underline{u}$ is the point in I with the smallest abscissa (line 19 of Listing 5.5), in order to make sure that AN_2 is obscured by the metal obstacle.

Finally, if both I and J are empty (line 20 of Listing 5.5) we look for the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$ and the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_2$. If the distances between both these pairs of points is below d_{Th} (line 24 of Listing 5.5), then I and J are redefined as the sets containing these two pairs of points and the localization is performed as in the first case. If the distance between the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$ is below d_{Th} and the distance between the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_2$ is above that threshold (line 26 of Listing 5.5), the TN position estimate is obtained

Listing 5.6: Pseudocode of procedure `findTargetPositionNoS2`.

```
1  procedure findTargetPositionNoS2
2
3  I ← {p_0, p_1} = C_0 ∩ C_1
4  if(I ≠ ∅)
5      u ← minAbscissa(I)
6  else if(I = ∅)
7      A ← C_0 ∩ R_01,  B ← C_1 ∩ R_01
8      I = {p_0, p_1} ← findNearer(A, B)
9      if (||p_0 − p_1|| ≤ d_Th)
10         u ← 1/2 (p_0 + p_1)
11     else
12         u ← [∞, ∞]
```

as the baricenter of the first pair of points. Similarly, if the distance between the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$ is above d_{Th} while the distance between the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_2$ is below d_{Th} (line 28 of Listing 5.5), the TN position estimate is obtained as the baricenter of the second pair of points. Otherwise, the TN position cannot be estimated.

Case 2. If $status_1 = 1$ and $status_2 = 0$ (line 16 of Listing 5.1) the localization algorithm is described in the procedure `findTargetPositionNoS2` in Listing 5.6. In this case, since the range estimate from AN_2 is not available, we can only consider $\mathscr{C}_0$ and $\mathscr{C}_1$ and their intersection I defined in (5.7).

If I is not empty (line 4 of Listing 5.6), the TN position estimate is the point in I with the smallest abscissa, for the same considerations about line 19 of Listing 5.5.

If I is empty (line 6 of Listing 5.6), we look for the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_1$ (as done in the case in line 8 of Listing 5.5). If their distance is below the distance threshold d_{Th} the TN position estimate is defined as the baricenter of such points. This choice is due to the fact that the distance estimate from AN_2 is not available and, therefore, we cannot rely on the set J (unlike in the case in line 8 of Listing 5.5). Otherwise, the available range estimates do not allow estimating the TN position.

Listing 5.7: Pseudocode of procedure `findTargetPositionNoS1`.

```
1  procedure findTargetPositionNoS1
2
3  J ← {q_0, q_1} = C_0 ∩ C_2
4  if(J ≠ ∅)
5      u ← maxAbscissa(J)
6  else if(J = ∅)
7      A ← C_0 ∩ R_02,  B ← C_2 ∩ R_02
8      J = {q_0, q_1} ← findNearer(A,B)
9      if (||q_0 − q_1|| ≤ d_Th)
10         u ← 1/2 (q_0 + q_1)
11     else
12         u ← [∞, ∞]
```

Case 3. If $status_1 = 0$ and $status_2 = 1$ (line 18 of Listing 5.1) the localization algorithm is described in the procedure `findTargetPositionNoS1` in Listing 5.7. In this case, we can only consider the intersection J of $\mathscr{C}_0$ and $\mathscr{C}_2$ defined in (5.7) since the range estimate from AN_1 is not available.

If J is not empty (line 4 of Listing 5.7), the TN position estimate is the point in J on the other side of AN_1 with respect to the metal obstacle, namely the one with the greatest abscissa.

If J is empty (line 6 of Listing 5.7) and the distance between the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_2$ is below the distance threshold d_{Th}, then the TN position estimate is defined as the baricenter of the two points. Otherwise, the TN position cannot be estimated.

Case 4. If $status_1 = 0$ and $status_2 = 0$ (line 20 of Listing 5.1) the range estimates from AN_1 and AN_2 are both unavailable. Hence, since only one distance estimate (namely, r_0) is known, the position estimate cannot be obtained. As a matter of fact, the knowledge of only one distance estimate does not allow implementing any localization strategy. The only available information is that the TN position estimates lie on a circumference centered in $\underline{s}_0$ with radius r_0.

Listing 5.8: Pseudocode of procedure `findTargetPositionVPeak`.

```
1 procedure findTargetPositionVPeak
2
3 if (vPeak_1 ≥ vPeak_Th ∧ vPeak_2 ≥ vPeak_Th)
4     findTargetPosition
5 else
6     if (vPeak_1 ≥ vPeak_2)
7         findTargetPositionNoS2
8     else
9         findTargetPositionNoS1
```

5.2.2 A Localization Algorithm Based on Distances and `vPeak`

In this subsection, an improved version of Algorithm 1 introduced in Subsection 5.2.1 is presented. In particular, the quality of the range estimates $\{r_i\}_{i=0}^{2}$ is evaluated in terms of the corresponding values of `vPeak` obtained by the RCMs. Since, as explained in Section 4.2, `vPeak` measures the RSS, large values of `vPeak` are likely relative to LoS paths while small values of `vPeak` are relative to NLoS paths. For this reason, denoting as $\{vPeak_i\}_{i=0}^{2}$ the values of `vPeak` relative to $\{\mathrm{AN}_i\}_{i=0}^{2}$, the greatest is $vPeak_i$, the most reliable is the distance estimate r_i. In the following, we refer to this algorithm as Algorithm 2.

The algorithm is the same as in Listing 5.1, except for the fact that in the localization phase the procedure `findTargetPosition` (line 14) is substituted by `findTargetPositionVPeak` explained in Listing 5.8. If both $vPeak_1$ and $vPeak_2$ are above a given threshold $vPeak_{\mathrm{Th}}$ (line 3 of Listing 5.8), then the localization is performed as in Listing 5.5. If $vPeak_1$ or $vPeak_2$ are below the threshold $vPeak_{\mathrm{Th}}$ (line 5 of Listing 5.8), then the localization is performed relying only on the range estimate from the node with the greatest `vPeak`, regardless of the fact that this is above or below the threshold $vPeak_{\mathrm{Th}}$. We remark that, if $vPeak_1$ and $vPeak_2$ are both below the threshold $vPeak_{\mathrm{Th}}$, we still consider the range estimates from the RCM with the largest value of `vPeak` (otherwise the TN position estimate could not be performed).

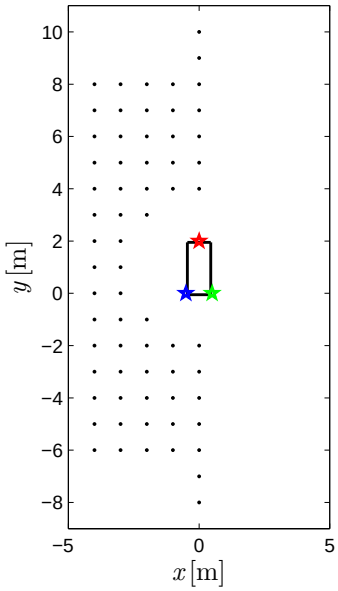


Figure 5.2: All the considered TN positions (*black dots*) are shown. The ANs (*coloured stars*) and the obstacle (*black rectangle*) are also shown.

5.3 Results Obtained with Three ANs

In this section, we show some results relative to the scenario described in Section 5.2. As in Figure 5.1, the values of the parameters L and V in (5.3) are set to 2 m and 0.5 m, respectively, so that the coordinates of the ANs, expressed in meters, are

$$\underline{s}_0 = [0, 2] \qquad \underline{s}_1 = [-0.5, 0] \qquad \underline{s}_2 = [0.5, 0]. \tag{5.10}$$

All the results are obtained with the distance threshold d_{Th} set to 0.5 m and the `vPeak` threshold $vPeak_{\mathrm{Th}}$ set to 5000. The choice of these parameters is due to experimental calibration.

We assume that range measurements from the three ANs are used to obtain the position estimates of a static TN. We consider 66 different TN positions which are shown in Figure 5.2.

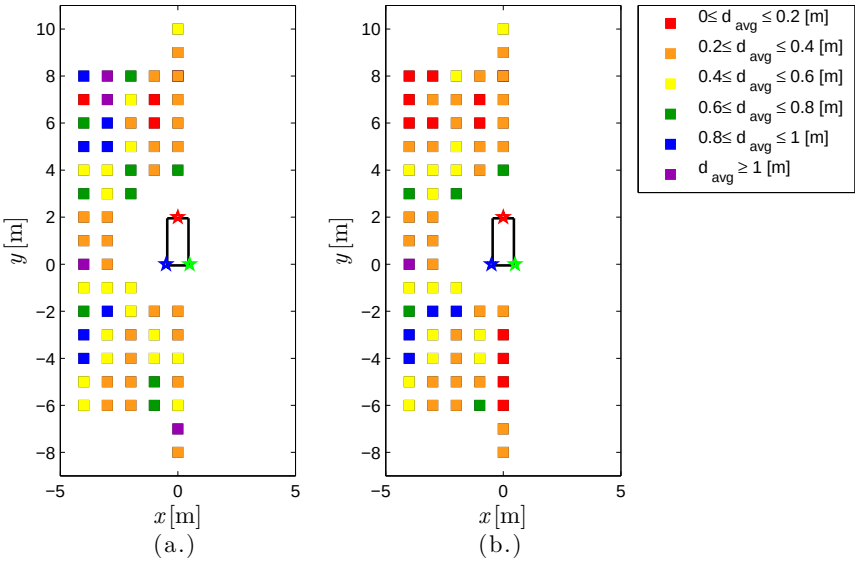


Figure 5.3: Values of d_{avg} relative to different TN positions for (a.) Algorithm 1 and (b.) Algorithm 2.

For each of the TN positions in Figure 5.2, Algorithm 1 described in Listing 5.1 is applied with $N_c = 100$. Once the 100 position estimates are given, the knowledge of the true TN position allows evaluating the average distance error d_{avg} defined in (5.2). Then, Algorithm 2 is applied to the same range estimates used with Algorithm 1 and the average distance error d_{avg} is evaluated for each TN position.

The results, in terms of d_{avg}, are shown in Figure 5.3. In particular, Figure 5.3 (a.) is relative to Algorithm 1. Each TN position is associated with a different colour, depending on the corresponding value of d_{avg}. More precisely: red squares correspond to values of d_{avg} smaller than 20 cm; orange squares correspond to values of d_{avg} between 20 cm and 40 cm; yellow squares correspond to values of d_{avg} between 40 cm and 60 cm; green squares correspond to values of d_{avg} between 60 cm and 80 cm; blue squares correspond to values of d_{avg} between 80 cm and 1 m; and violet squares correspond to values of d_{avg} larger than 1 m. Figure 5.3 (b.) can be similarly interpreted but for the fact that it refers to the results obtained with Algorithm 2.

From the results in Figure 5.3 one can deduce that Algorithm 2 is, indeed, an improvement of Algorithm 1. As a matter of fact, Figure 5.3 (a.) shows that when using Algorithm 1, the value of d_{avg} is below 20 cm only in 3 (over 66) TN positions, corresponding to 4.5% of the cases. When using Algorithm 2, instead, the value of d_{avg} is below 20 cm in 11 TN positions, corresponding to 16% of the cases, namely 4 times the percentage obtained with Algorithm 1.

Furthermore, d_{avg} obtained with Algorithm 1 is below 40 cm in 28 TN positions, as shown in Figure 5.3 (a.). This corresponds to a percentage of 42%, which is nearly 10 times the percentage relative to d_{avg} smaller than 20 cm. However, in more than a half of the considered TN positions, the average distance error d_{avg} is greater than 40 cm. At the opposite, when using Algorithm 2, the value of d_{avg} is below 40 cm in 40 TN positions, namely in 60% of the cases.

Moreover, when using Algorithm 1, the value of d_{avg} is larger than 1 m in 4 TN positions, while if Algorithm 2 is used this happens only once.

Even if, generally speaking, Algorithm 2 improves the performance of Algorithm 1, a comparison between Figure 5.3 (a.) and Figure 5.3 (b.) shows that there is one TN position where the value of d_{avg} obtained with Algorithm 2 is greater than that obtained with Algorithm 1. This happens when the TN coordinates, expressed in meters, are $\underline{u} = [-2, -2]$. This is due to the fact that, in this case, some range estimates are accurate even if they are associated with small values of `vPeak`. Algorithm 2 ignores such range estimates, leading to a performance degradation with respect to Algorithm 1.

However, in many TN positions, the value of d_{avg} obtained with Algorithm 1 is analogous to that obtained with Algorithm 2 (for instance, in the TN positions on the left of the metal obstacle) and in 16 TN positions the value of d_{avg} obtained with Algorithm 2 is smaller than that obtained with Algorithm 1. The improved performance of Algorithm 2 is particularly evident in some TN positions.

As an illustrative example, let us consider the TN position whose coordinates, expressed in meters, are $\underline{u} = [-3, 8]$. In this case, Figure 5.3 shows that the improvement of Algorithm 2 is particularly noticeable as it allows reducing d_{avg} below 20 cm while, according to Algorithm 1, d_{avg} is larger than 1 m.

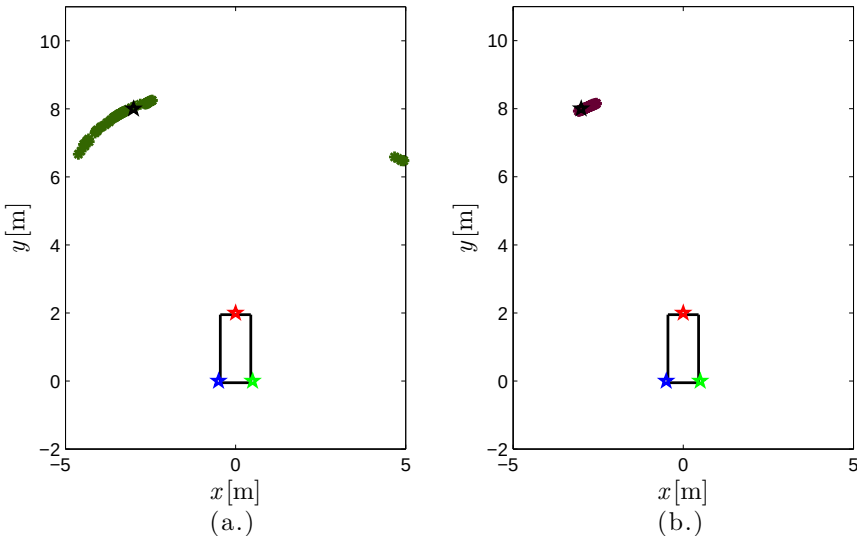


Figure 5.4: TN position estimates with (a.) Algorithms 1 and (b.) 2.

All the 100 position estimates are shown in Figure 5.4, where (a.) shows the position estimates obtained with Algorithm 1 and (b.) shows the position estimates obtained with Algorithm 2. From Figure 5.4 (a.) it can be observed that the position estimates with Algorithm 1 can be very inaccurate. To be precise, 16 position estimates (among the obtained 100) are nearly 9 meters from the true TN position. This is due to the fact that, in this scenario, AN_2 is in NLoS with the TN, leading to wrong estimates of $\hat{r}_2$ which can have a big impact on the final position estimate.

An illustrative example is shown in Figure 5.5, where the circumferences $\{\mathscr{C}_i\}_{i=0}^{2}$ centered in $\{\underline{s}_i\}_{i=0}^{2}$ with radii $\{r_i\}_{i=0}^{2}$ are shown (dashed lines) together with the circumferences $\{\hat{\mathscr{C}}_i^{(29)}\}_{i=0}^{2}$ obtained with the range estimates $\{\hat{r}_i^{(29)}\}_{i=0}^{2}$ in the $29-$th iteration. It can be observed that the estimates of r_0 and r_1 are sufficiently accurate and the TN is near one of the two points in $\hat{\mathscr{C}}_0^{(29)} \cap \hat{\mathscr{C}}_1^{(29)}$. At the opposite, r_2 is underestimated by nearly 1 m. Due to this fact, the two points in which $\hat{\mathscr{C}}_2^{(29)}$ intersects $\hat{\mathscr{C}}_0^{(29)}$ are far from the true TN position and, moreover, one of them is near to the wrong intersection between $\hat{\mathscr{C}}_0^{(29)}$ and $\hat{\mathscr{C}}_1^{(29)}$. Referring to the procedure `findTargetPosition`, we are in the case shown in line 5. When looking for the

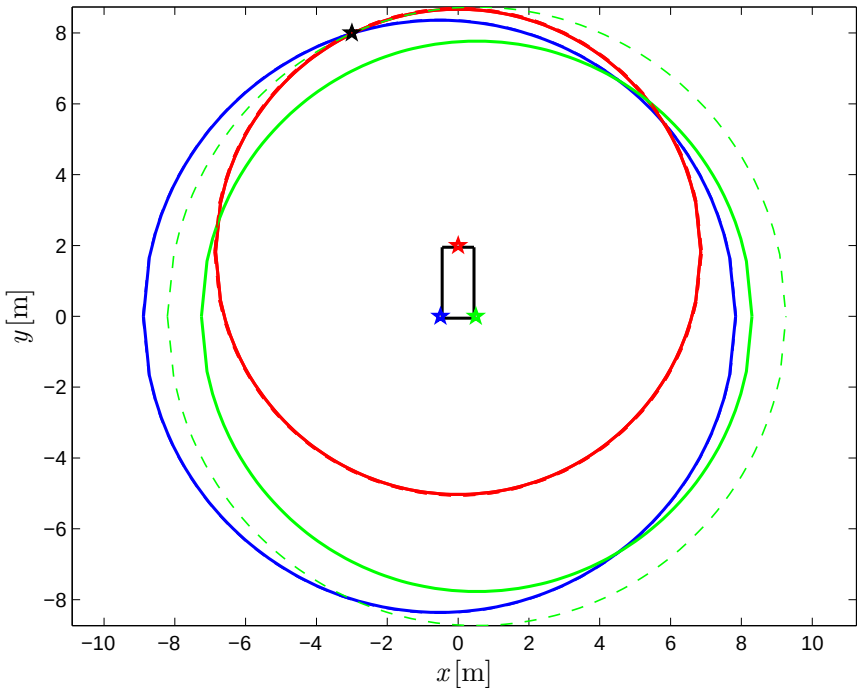


Figure 5.5: The circumferences $\{\mathscr{C}_i\}_{i=0}^{2}$ (*dashed lines*) and their estimates obtained from the range estimates in the $29-$th iteration $\{\hat{\mathscr{C}}_i^{(29)}\}_{i=0}^{2}$ (*solid lines*) are shown. The TN position estimate in this case, expressed in meters, is $[5.73, 5.63]$.

two nearest points of I and J we find that they are the ones on the right of the figure, which turn out to be the wrong ones. The procedure `findTargetPositionVPeak` used in Algorithm 2 can overcome this problem. As a matter of fact, in the $29-$th iteration the value of $vPeak_2$ is below the threshold $vPeak_{\mathrm{Th}}$, so that the localization estimate is performed according to the procedure `findTargetPositionNoS2`, i.e., ignoring the range estimate from AN_2 and choosing as a final position estimate the point in $\hat{\mathscr{C}}_0^{(29)} \cap \hat{\mathscr{C}}_1^{(29)}$ with the smallest abscissa, namely the one which is actually nearer to the TN.

The previous example shows how the values of $\{vPeak_i\}_{i=0}^{2}$ can be useful to improve the localization performance. These values in the considered TN positions are

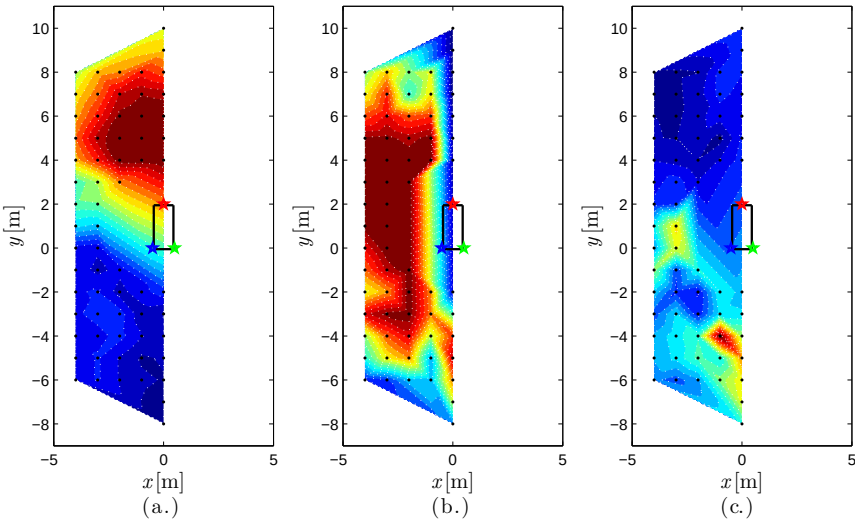


Figure 5.6: The values of (a.) $vPeak_0$, (b.) $vPeak_1$, and (c.) $vPeak_2$ are shown.

shown in Figure 5.6. More precisely, Figure 5.6 (a.) refers to $vPeak_0$, Figure 5.6 (b.) refers to $vPeak_1$, and Figure 5.6 (c.) refers to $vPeak_2$. Figure 5.6 (a.) shows that, as expected, if we consider TN positions with ordinate greater than 2 m (so that the TN is in LoS with AN_0), the values of $vPeak_0$ are large and they gradually decrease as the distance r_0 between the TN and AN_0 increases. At the opposite, if the obstacle is between the TN and AN_0, the values of $vPeak_0$ are small because of the signal attenuation due to the presence of the metal obstacle. Concerning $vPeak_1$, Figure 5.6 (b.) shows that its values are often large, since the majority of the TN positions are on the same side of the metal obstacle with respect to AN_1. At the opposite, Figure 5.6 (c.) shows that the values of $vPeak_2$ are often small, especially in correspondence to TN positions with ordinate greater than 2 m. Once again, this is due to the presence of the metal obstacle between the TN and AN_2.

Figure 5.6 shows that three ANs are not sufficient to have a good coverage, in terms of `vPeak`, in all the considered TN positions. For this reason, aiming at improving the results in Figure 5.3, in next sections we consider a similar scenario with four ANs.

5.4 Algorithm with Four ANs

In this section, aiming at improving the performance obtained with the localization setup described in Section 5.2, four ANs are considered. Defining two parameters L and V, we assume that their coordinates are

$$\underline{s}_0 = [-V, L] \qquad \underline{s}_1 = [V, L] \qquad \underline{s}_2 = [-V, 0] \qquad \underline{s}_3 = [-V, L]. \tag{5.11}$$

The considered scenario is shown in Figure 5.7, where the black rectangle represents the metal obstacle, the four ANs are denoted with coloured stars, and the values of the parameters L and V in (5.11) are set to 2 m and 0.5 m, respectively.

The knowledge of the coordinates of the four ANs and of the true distances $\{r_i\}_{i=0}^3$ between the TN and the $i-$th AN allows deriving the circumferences $\{\mathscr{C}_i\}_{i=0}^3$ centered in $\{\underline{s}_i\}_{i=0}^3$

$$\begin{cases} (x+V)^2 + (y-L)^2 = r_0^2 \\ (x+V)^2 + y^2 = r_1^2 \\ (x-V)^2 + y^2 = r_2^2 \\ (x-V)^2 + (y-L)^2 = r_3^2 \end{cases} \tag{5.12}$$

the intersection of which corresponds to the TN position. Since the true distances are supposed to be unknown, the TN localization in the $j-$th iteration can only be based on their estimates, hence on the set of circumferences $\{\hat{\mathscr{C}}_i^{(j)}\}_{i=0}^3$

$$\begin{cases} (x+V)^2 + (y-L)^2 = [\hat{r}_0^{(j)}]^2 \\ (x+V)^2 + y^2 = [\hat{r}_1^{(j)}]^2 \\ (x-V)^2 + y^2 = [\hat{r}_2^{(j)}]^2 \\ (x-V)^2 + (y-L)^2 = [\hat{r}_3^{(j)}]^2 \end{cases} \tag{5.13}$$

centered in $\{\underline{s}_i\}_{i=0}^3$ and whose radii are the estimated distances $\{\hat{r}_i^{(j)}\}_{i=0}^3$.

In the remainder of this section, the two localization algorithms described in Section 5.2 are rearranged to the case of four ANs. The algorithm details are given in the two following subsections. As in Section 5.2, in the pseudocode the superscript (j) is omitted and we simplify the notation writing $[\star]$ instead of $[\hat{\star}]$ even if the considered quantities are estimates.

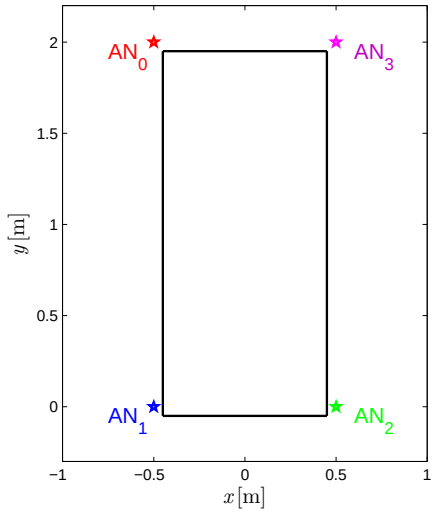


Figure 5.7: The ANs (*coloured stars*) and the metal obstacle in the second scenario (*black rectangle*) are shown.

5.4.1 A Distance-based Localization Algorithm

In this subsection, Algorithm 1 introduced in Subsection 5.2.1 is extended to the case of four ANs. A pseudocode of the algorithm implemented on the host (namely, the Raspberry Pi connected to the four RCMs) is shown in Listing 5.9.

The first part of the algorithm involves the acquisition of the distance estimates between each AN and a TN. The first range request is performed (in broadcast) by AN_0 (line 5 of Listing 5.9) and, as soon as a TN replies, the remaining ANs directly enquire (in sequence) that TN (lines 8, 12, and 16 of Listing 5.9). If, due to the presence of a metal obstacle in the considered scenario, the range requests are not successful, the values of $status_1$, $status_2$ and/or $status_3$ are kept equal to 0, otherwise they are set to 1 and the distance estimates $\{r_i\}_{i=1}^{3}$ and the values of $\{vPeak_i\}_{i=1}^{3}$ are saved. The values of $\{vPeak_i\}_{i=1}^{3}$ are saved because, even if we do not use them to improve the localization accuracy, we rely on them to select which triplet of range estimates must be used if all the four estimates are available.

Listing 5.9: Pseudocode of Algorithm 1 for four ANs.

```
1  procedure algorithm 1
2
3  for cycle ∈ {1,...,N_c}
4      status_1 ← 0,  status_2 ← 0,  status_3 ← 0
5      rcmRangeToEx(AN_0, BROADCAST_NODE_ID)
6      r_0 ← rangeInfo.precisionRangeMm
7      vPeak_0 ← rangeInfo.vPeak
8      if (rcmRangeToEx(AN_1, rangeInfo.responderId)≠ERR)
9          status_1 ← 1
10         r_1 ← rangeInfo.precisionRangeMm
11         vPeak_1 ← rangeInfo.vPeak
12     if (rcmRangeToEx(AN_2, rangeInfo.responderId)≠ERR)
13         status_2 ← 1
14         r_2 ← rangeInfo.precisionRangeMm
15         vPeak_2 ← rangeInfo.vPeak
16     if (rcmRangeToEx(AN_3, rangeInfo.responderId)≠ERR)
17         status_3 ← 1
18         r_3 ← rangeInfo.precisionRangeMm
19         vPeak_3 ← rangeInfo.vPeak
20
21     if (status_1 = 1 ∧ status_2 = 1 ∧ status_3 = 1)
22         findTargetPosition4
23     else if (status_1 = 1 ∧ status_2 = 1 ∧ status_3 = 0)
24         findTargetPositionNoS3
25     else if (status_1 = 1 ∧ status_2 = 0 ∧ status_3 = 1)
26         findTargetPositionNoS2
27     else if (status_1 = 0 ∧ status_2 = 1 ∧ status_3 = 1)
28         findTargetPositionNoS1
29     else if (status_1 = 1 ∧ status_2 = 0 ∧ status_3 = 0)
30         findTargetPositionS0S1
31     else if (status_1 = 0 ∧ status_2 = 1 ∧ status_3 = 0)
32         findTargetPositionS0S2
33     else if (status_1 = 0 ∧ status_2 = 0)
34         u ← [∞,∞]
```

Listing 5.10: Pseudocode of function `maxOrdinate`.

```
1 function u = maxOrdinate(I)
2 input: I = {p_0 = [x_0,y_0], p_1 = [x_1,y_1]} coordinates of two points
3 output: u chosen point
4
5 if (y_0 < y_1)
6     u ← p_1
7 else
8     u ← p_0
```

Listing 5.11: Pseudocode of function `minOrdinate`.

```
1 function u = minOrdinate(I)
2 input: I = {p_0 = [x_0,y_0], p_1 = [x_1,y_1]} coordinates of two points
3 output: u chosen points
4
5 if (y_0 < y_1)
6     u ← p_0
7 else
8     u ← p_1
```

As a matter of fact, even if four range estimates are available, at least one of them is not reliable because in the considered scenario at least one AN is in NLoS with the TN. To be coherent with the considerations made in Section 5.2 we assume that the most inaccurate range estimate is the one with the smallest `vPeak`.

As in the case with three ANs, the second part of the algorithm is dedicated to the TN localization, on the basis of the parameters obtained in the first part. Depending on the number of available range estimates, different localization strategies can be considered.

In these localization strategies, the functions `findNearer`, `maxAbscissa`, and `minAbscissa` defined in Listing 5.2, Listing 5.3, and Listing 5.4, respectively, are used. Moreover, two additional functions are used in the following.

Listing 5.12: Pseudocode of procedure `findTargetPosition4`.

```
1  procedure findTargetPosition4
2
3  if (vPeak_0 ≤ vPeak_1 ∧ vPeak_0 ≤ vPeak_2 ∧ vPeak_0 ≤ vPeak_3)
4      findTargetPositionNoS0
5  else if (vPeak_1 ≤ vPeak_2 ∧ vPeak_1 ≤ vPeak_3 ∧ vPeak_1 ≤ vPeak_0)
6      findTargetPositionNoS1
7  else if (vPeak_2 ≤ vPeak_1 ∧ vPeak_2 ≤ vPeak_3 ∧ vPeak_2 ≤ vPeak_0)
8      findTargetPositionNoS2
9  else if (vPeak_3 ≤ vPeak_1 ∧ vPeak_3 ≤ vPeak_2 ∧ vPeak_3 ≤ vPeak_0)
10     findTargetPositionNoS3
```

The function `maxOrdinate` in Listing 5.10 takes as input a set containing the coordinates of two points and returns the coordinate of the point with the greatest ordinate.

Analogously, the function `minOrdinate` defined in Listing 5.11 takes as input a set containing the coordinates of two points and returns the coordinate of the point with the smallest ordinate.

We can now describe the position estimate phase in Listing 5.9. Eight cases are identified on the basis of the number of available distance estimates and of the values of `vPeak`.

Case 1. If $status_1 = 1$, $status_2 = 1$, and $status_3 = 1$ (line 21 of Listing 5.9) the localization algorithm is described in `findTargetPosition4` in Listing 5.12. Observe that in the considered scenario, shown in Figure 5.7, for each possible TN position at least one AN is in NLoS with the TN, due to the presence of the metal obstacle. For this reason, even if four range estimates are available, in the proposed algorithm one of them is always discarded. More precisely, we discard the range estimate associated with the smallest `vPeak`, as it is likely associated with a NLoS path, and, consequently, can be far inaccurate. Hence, in Listing 5.12 the position estimate is made according to the procedures described in Listings 5.13-5.17, depending on the values of `vPeak`.

Listing 5.13: Pseudocode of procedure `findTargetPositionNoS0`.

```
1  procedure findTargetPositionNoS0
2
3  I ← {p_0, p_1} = C_1 ∩ C_3
4  J ← {q_0, q_1} = C_1 ∩ C_2
5  if(I ≠ ∅ ∧ J ≠ ∅)
6      {p, q} ← findNearer(I,J)
7      u ← 1/2 (p + q)
8  else if(I = ∅ ∧ J ≠ ∅)
9      A ← C_1 ∩ R_13,  B ← C_3 ∩ R_13
10     I = {p_0, p_1} ← findNearer(A,B)
11     if (||p_0 − p_1|| ≤ d_Th)
12         {p, q} ← findNearer(I,J)
13         u ← 1/2 (p + q)
14     else
15         u ← maxAbscissa(J)
16 else if (J = ∅)
17     findTargetPositionS1S3
```

Case 2. If $status_1 = 1$, $status_2 = 1$, and $status_3 = 1$ and assuming that $vPeak_0$ is the minimum value of `vPeak` (line 3 of Listing 5.12), the localization algorithm is described in the procedure `findTargetPositionNoS0` in Listing 5.13.

Ignoring the range measurement from AN_0, we can consider the sets

$$I = \mathscr{C}_1 \cap \mathscr{C}_3 \qquad J = \mathscr{C}_1 \cap \mathscr{C}_2. \tag{5.14}$$

If both I and J are not empty (line 5 of Listing 5.13), the function `findNearer` defined in Listing 5.2 is used to obtain the two nearest points of I and J. The TN position estimate is then found as the baricenter of these two points.

If $\mathscr{C}_1$ and $\mathscr{C}_2$ intersect but I is empty (line 8 of Listing 5.13), we look for the two nearest points of $\mathscr{C}_1$ and $\mathscr{C}_3$, which can be obtained by intersecting both such circumferences with the line $\mathscr{R}_{13}$ which goes through their centers, namely

$$\mathscr{R}_{13} : x = V. \tag{5.15}$$

Listing 5.14: Pseudocode of procedure `findTargetPositionS1S3`.

```
1  procedure findTargetPositionS1S3
2
3  if(I ≠ ∅)
4      u ← maxAbscissa(I)
5  else if(I = ∅)
6      A ← C1 ∩ R13,  B ← C3 ∩ R13
7      I = {p0, p1} ← findNearer(A,B)
8      if (||p0 − p1|| ≤ dTh)
9          u ← 1/2 (p0 + p1)
10     else
11         u ← [∞, ∞]
```

The two nearest points from $\mathscr{C}_1 \cap \mathscr{R}_{13}$ and $\mathscr{C}_3 \cap \mathscr{R}_{13}$ can be found using the function defined in Listing 5.2. If their distance is below a given threshold d_{Th} (line 11 of Listing 5.13) I is redefined as the set containing the two points and the localization is performed as in the first case. Otherwise, $\underline{u}$ is defined as the point in J which is on the other side of the metal obstacle with respect to AN_0, namely the one with the greatest abscissa (line 14 of Listing 5.13). As a matter of fact, since $vPeak_0$ is the smallest value of `vPeak`, it is reasonable to assume that the metal obstacle is between AN_0 and the TN.

If $\mathscr{C}_1$ and $\mathscr{C}_2$ do not intersect, so that J is empty (line 16 of Listing 5.13), the localization is performed using the procedure `findTargetPositionS1S3` defined in Listing 5.14.

If the two circumferences $\mathscr{C}_1$ and $\mathscr{C}_3$ intersect, the two obtained points are symmetric with respect to the line $\mathscr{R}_{13}$ defined in (5.15). The TN position estimate is chosen as the point on the same side of the metal obstacle as AN_1 and AN_3, namely the one with the greatest abscissa. Instead, if $\mathscr{C}_1$ and $\mathscr{C}_3$ do not intersect the two nearest points of such circumferences are found and if their distance is below the threshold d_{Th} the TN position is estimated as their baricenter. Otherwise, the position estimate cannot be performed.

Listing 5.15: Pseudocode of procedure `findTargetPositionNoS1`.

```
1  procedure findTargetPositionNoS1
2
3  I ← {p_0, p_1} = C_0 ∩ C_2
4  J =← {q_0, q_1} = C_0 ∩ C_3
5  if(I ≠ ∅ ∧ J ≠ ∅)
6      {p, q} ← findNearer(I,J)
7      u ← 1/2 (p + q)
8  else if(I = ∅ ∧ J ≠ ∅)
9      A ← C_0 ∩ R_02, B ← C_2 ∩ R_02
10     I = {p_0, p_1} ← findNearer(A,B)
11     if (||p_0 − p_1|| ≤ d_Th)
12         {p, q} ← findNearer(I,J)
13         u ← 1/2 (p + q)
14     else
15         u ← minAbscissa(J)
16 else if(J = ∅)
17     findTargetPositionS0S2
```

Case 3. If $status_1 = 0$, $status_2 = 1$, and $status_3 = 1$ (line 27 of Listing 5.9), or if $status_1 = 1$, $status_2 = 1$, $status_3 = 1$, and $vPeak_1$ is the minimum value of `vPeak` (line 5 of Listing 5.12) the localization algorithm is described in the procedure `findTargetPositionNoS1` in Listing 5.15.

In this case, the range measurements from AN_1 is ignored (either because it is unknown or because it is the one associated with the smallest `vPeak` in case that all the range estimates are available). We can then define

$$I = \mathscr{C}_0 \cap \mathscr{C}_2 \qquad J = \mathscr{C}_0 \cap \mathscr{C}_3 \tag{5.16}$$

as in lines 3 and 4 of Listing 5.7.

If both pairs of circumferences intersect (line 5), the two nearest points from I and J are found using the function `findNearer` defined in Listing 5.2 and the TN position estimate is the baricenter of these two points.

Listing 5.16: Pseudocode of procedure `findTargetPositionNoS2`.

```
1  procedure findTargetPositionNoS2
2
3  I ← {p_0, p_1} = C_1 ∩ C_3
4  J ← {q_0, q_1} = C_0 ∩ C_3
5  if(I ≠ ∅ ∧ J ≠ ∅)
6      {p, q} ← findNearer(I,J)
7      u ← 1/2 (p + q)
8  else if(I = ∅ ∧ J ≠ ∅)
9      A ← C_1 ∩ R_13,  B ← C_3 ∩ R_13
10     I = {p_0, p_1} ← findNearer(A,B)
11     if (||p_0 − p_1|| ≤ d_Th)
12         {p, q} ← findNearer(I,J)
13         u ← 1/2 (p + q)
14     else
15         u ← maxAbscissa(J)
16 else if(J = ∅)
17     findTargetPositionS1S3
```

If $\mathscr{C}_0$ and $\mathscr{C}_3$ intersect but I is empty (line 8 of Listing 5.7), the two nearest points of $\mathscr{C}_0$ and $\mathscr{C}_2$ are found by intersecting these circumferences with the line $\mathscr{R}_{02}$ which goes through their centers

$$\mathscr{R}_{02} : x = -V. \tag{5.17}$$

If the distance between the two points is below a given threshold d_{Th} (line 11 of Listing 5.7) I is redefined as the set containing these two points and the localization is performed as in the first case. Otherwise, it is reasonable to assume that the TN position estimate corresponds to the point in J with the smallest abscissa, so that it is on the other side of the metal obstacle with respect to the AN associated with the lowest `vPeak`, namely AN_1 (line 14 of Listing 5.7).

If $\mathscr{C}_0$ and $\mathscr{C}_3$ do not intersect, so that J is empty, the localization is performed using the procedure `findTargetPositionS0S2` defined later in Listing 5.19.

Listing 5.17: Pseudocode of procedure `findTargetPositionNoS3`.

```
1  procedure findTargetPositionNoS3
2
3  I ← {p_0, p_1} = C_0 ∩ C_2
4  J ← {q_0, q_1} = C_1 ∩ C_2
5  if(I ≠ ∅ ∧ J ≠ ∅)
6       {p, q} ← findNearer(I,J)
7       u ← 1/2 (p + q)
8  else if(I = ∅ ∧ J ≠ ∅)
9       A ← C_0 ∩ R_02,  B ← C_2 ∩ R_02
10      I = {p_0, p_1} ← findNearer(A,B)
11      if (||p_0 − p_1|| ≤ d_Th)
12           {p, q} ← findNearer(I,J)
13           u ← 1/2 (p + q)
14      else
15           u ← minAbscissa(J)
16 else if(J = ∅)
17      findTargetPositionS0S2
```

Case 4. If $status_1 = 1$, $status_2 = 0$, and $status_3 = 1$ (line 25 of Listing 5.9), or if $status_1 = 1$, $status_2 = 1$ $status_3 = 1$, and $vPeak_2$ is the minimum value of `vPeak` (line 7 of Listing 5.12) the localization algorithm is described in the procedure `findTargetPositionNoS2` in Listing 5.16.

The localization scheme is analogous to `findTargetPositionNoS0` implemented in in Listing 5.13 and can be obtained from it by substituting $\mathscr{C}_1$ with $\mathscr{C}_3$, $\mathscr{C}_2$ with $\mathscr{C}_0$, and $\mathscr{C}_3$ with $\mathscr{C}_1$.

Case 5. If $status_1 = 1$, $status_2 = 1$, and $status_3 = 0$ (line 23 of Listing 5.9), or if $status_1 = 1$, $status_2 = 1$, $status_3 = 1$, and $vPeak_3$ is the minimum value of `vPeak`, (line 9 of Listing 5.12) the localization algorithm is described in the procedure `findTargetPositionNoS3` in Listing 5.17.

The localization can be derived from `findTargetPositionNoS1` defined in Listing 5.15 by substituting $\mathscr{C}_0$ with $\mathscr{C}_2$, $\mathscr{C}_2$ with $\mathscr{C}_0$, and $\mathscr{C}_3$ with $\mathscr{C}_1$.

Listing 5.18: Pseudocode of procedure `findTargetPositionS0S1`.

```
1  procedure findTargetPositionS0S1
2
3  I ← {p_0, p_1} = C_0 ∩ C_1
4  if(I ≠ ∅)
5      u ← maxOrdinate(I)
6  else
7      A ← C_0 ∩ R_01,  B ← C_1 ∩ R_01
8      I = {p_0, p_1} ← findNearer(A, B)
9      if (||p_0 − p_1|| ≤ d_Th)
10         u ← 1/2 (p_0 + p_1)
11     else
12         u ← [∞, ∞]
```

Case 6. If $status_1 = 1$, $status_2 = 0$, and $status_3 = 0$ (line 29 of Listing 5.9) the procedure `findTargetPositionS0S1` described in Listing 5.18 is used to perform the localization. In this case, since the range estimates from AN_2 and AN_3 are not available, only two circumferences can be derived, namely $\mathscr{C}_0$ and $\mathscr{C}_1$.

If they intersect, the two obtained points have the same abscissa and different ordinates since they are symmetric with respect to the line $\mathscr{R}_{01}$ which goes through the centers of the circumferences, whose equation is

$$\mathscr{R}_{01} : y = L. \tag{5.18}$$

Since the range requests from AN_2 and AN_3 were not successful, it is reasonable to assume that the TN position can be estimated as the point with the greatest ordinate, namely the one further from AN_2 and AN_3 and nearer to AN_0 and AN_1.

If $\mathscr{C}_0$ and $\mathscr{C}_1$ do not intersect, we look for the two nearest points of such circumferences. If their distance is below the threshold d_{Th} the TN position is defined as the baricenter of the two points. Otherwise, the available range estimates do not give enough information to estimate the TN position and the localization cannot be performed.

Listing 5.19: Pseudocode of procedure `findTargetPositionS0S2`.

```
1  procedure findTargetPositionS0S2
2
3  I ← {p_0, p_1} = C_0 ∩ C_2
4  if(I ≠ ∅)
5      u ← minAbscissa(I)
6  else
7      A ← C_0 ∩ R_02,  B ← C_2 ∩ R_02
8      I = {p_0, p_1} ← findNearer(A, B)
9      if (||p_0 − p_1|| ≤ d_Th)
10         u ← 1/2 (p_0 + p_1)
11     else
12         u ← [∞, ∞]
```

Case 7. If $status_1 = 0$, $status_2 = 1$, and $status_3 = 0$ (line 31 of Listing 5.9) the localization algorithm is described in the procedure `findTargetPositionS0S2` in Listing 5.19, which is similar to `findTargetPositionS0S1` in Listing 5.18. This procedure is also used in Listing 5.15 and Listing 5.17 if $J = \emptyset$.

If the two circumferences $\mathscr{C}_0$ and $\mathscr{C}_2$ intersect, the two obtained points are symmetric with respect to the line $\mathscr{R}_{02}$ which goes through their centers, defined in (5.17). The TN position is then estimated as the point with the smallest abscissa, which is, for the geometry of the problem, the point on the same side of the metal obstacle as AN_0 and AN_2.

If $\mathscr{C}_0$ and $\mathscr{C}_2$ do not intersect, we look for the two nearest points of them and if their distance is below the threshold d_{Th} the TN position is estimated as their baricenter. Otherwise, the position estimate cannot be performed since the available range estimates do not give enough information.

Case 8. If $status_1 = 0$, $status_2 = 0$ (line 33 of Listing 5.9) the TN position cannot be estimated, regardless of $status_3$. Even if $status_3 = 1$ and $\mathscr{C}_0 \cap \mathscr{C}_3 \neq \emptyset$ we have no sufficient information to decide which of the two intersection points corresponds to the TN position estimate.

Listing 5.20: Pseudocode of procedure `findTargetPosition4VPeak`.

```
1  procedure findTargetPosition4VPeak
2
3  if (vPeak_0 ≤ vPeak_1 ∧ vPeak_0 ≤ vPeak_2 ∧ vPeak_0 ≤ vPeak_3)
4      findTargetPositionNoS0VPeak
5  else if (vPeak_1 ≤ vPeak_2 ∧ vPeak_1 ≤ vPeak_3 ∧ vPeak_1 ≤ vPeak_0)
6      findTargetPositionNoS1VPeak
7  else if (vPeak_2 ≤ vPeak_1 ∧ vPeak_2 ≤ vPeak_3 ∧ vPeak_2 ≤ vPeak_0)
8      findTargetPositionNoS2VPeak
9  else if (vPeak_3 ≤ vPeak_1 ∧ vPeak_3 ≤ vPeak_2 ∧ vPeak_3 ≤ vPeak_0)
10     findTargetPositionNoS3VPeak
```

5.4.2 A Localization Algorithm Based on Distances and `vPeak`

In this subsection, the analogous of the algorithm in Subsection 5.2.2 is explained. It is an improved version of the algorithm explained in the previous subsection, where the values $\{vPeak_i\}_{i=0}^{3}$ which correspond to the `vPeak` of $\{\mathrm{AN}_i\}_{i=0}^{3}$ are used to analyze the quality of the range estimates $\{r_i\}_{i=0}^{3}$.

The distance acquisition step in the algorithm is the same as in Listing 5.9, while in the position estimate step some procedure are changed in order to take into account the values of `vPeak`.

In particular, if $status_1 = 1$, $status_2 = 1$, and $status_3 = 1$, we substitute the procedure `findTargetPosition4` by the procedure `findTargetPosition4VPeak` defined in Listing 5.20, where the four procedures `findTargetPositionNoS`$\star$ are changed into `findTargetPositionNoS`$\star$`VPeak` defined in the respective listings.

In the procedure `findTargetPositionNoS0VPeak`, defined in Listing 5.21, if all the values of $\{vPeak\}_{i=1}^{3}$ are above the threshold $vPeak_{\mathrm{Th}}$ (line 3 of Listing 5.21), then the position estimate is performed according to `findTargetPositionNoS0` as in Algorithm 1. If $vPeak_1$ is below the threshold $vPeak_{\mathrm{Th}}$ and it is the smallest value of `vPeak` (line 5 of Listing 5.21), the position estimate is performed relying only on the distance estimates from AN_2 and AN_3, according to the procedure

Listing 5.21: Pseudocode of procedure `findTargetPositionNoS0VPeak`.

```
1  procedure findTargetPositionNoS0VPeak
2
3  if (vPeak_1 ≥ vPeak_Th ∧ vPeak_2 ≥ vPeak_Th ∧ vPeak_3 ≥ vPeak_Th)
4      findTargetPositionNoS0
5  else if (vPeak_1 < vPeak_2 ∧ vPeak_1 < vPeak_3)
6      I ← {p_0, p_1} = C_2 ∩ C_3
7      findTargetPositionS2S3
8  else if (vPeak_2 < vPeak_1 ∧ vPeak_2 < vPeak_3)
9      I ← {p_0, p_1} = C_1 ∩ C_3
10     findTargetPositionS1S3
11 else if (vPeak_3 < vPeak_1 ∧ vPeak_3 < vPeak_2)
12     I ← {p_0, p_1} = C_1 ∩ C_2
13     if(I ≠ ∅)
14         u ← maxAbscissa(I)
15     else
16         u ← [∞, ∞]
```

`findTargetPositionS2S3` defined in Listing 5.25. Similarly, if $vPeak_2$ is below $vPeak_{\text{Th}}$ and it is the smallest value of `vPeak` (line 8 of Listing 5.21), the position estimate is performed using the procedure `findTargetPositionS1S3` defined in Listing 5.14. Finally, if $vPeak_3$ is below $vPeak_{\text{Th}}$ and it is the smallest value of `vPeak` (line 11 of Listing 5.21) the localization estimate is performed using only the range measurements from AN_1 and AN_2. If the associated circumferences intersect, the position estimate is chosen as the point with the greatest abscissa (i.e., the one further from AN_0), otherwise the position estimate cannot be obtained.

In procedure `findTargetPositionNoS1VPeak` in Listing 5.22 the localization scheme is the same as the one in the procedure `findTargetPositionNoS0VPeak`, beside the fact that now the unavailable or ignored range estimate is the one relative to AN_1. If all the values of `vPeak` are above the threshold $vPeak_{\text{Th}}$ (line 3 of Listing 5.22), then the position estimate is performed as in Algorithm 1 according to `findTargetPositionNoS1`. If $vPeak_0$ is below the threshold $vPeak_{\text{Th}}$ and it is

Listing 5.22: Pseudocode of procedure `findTargetPositionNoS1VPeak`.

```
1  procedure findTargetPositionNoS1VPeak
2
3  if (vPeak_0 ≥ vPeak_Th ∧ vPeak_2 ≥ vPeak_Th ∧ vPeak_3 ≥ vPeak_Th)
4      findTargetPositionNoS1
5  else if (vPeak_0 < vPeak_2 ∧ vPeak_0 < vPeak_3)
6      I ← {p_0, p_1} = C_2 ∩ C_3
7      findTargetPositionS2S3
8  else if (vPeak_2 < vPeak_0 ∧ vPeak_2 < vPeak_3)
9      I ← {p_0, p_1} = C_0 ∩ C_3
10     if(I ≠ ∅)
11         u ← minAbscissa(I)
12     else
13         u ← [∞, ∞]
14 else if (vPeak_3 < vPeak_0 ∧ vPeak_3 < vPeak_2)
15     I ← {p_0, p_1} = C_0 ∩ C_2
16     findTargetPositionS0S2
```

smaller than $vPeak_2$ and $vPeak_3$ (line 5 of Listing 5.22) the TN position estimate is obtained using `findTargetPositionS2S3`. If $vPeak_2$ is below the threshold $vPeak_{\text{Th}}$ and it is the smallest value of `vPeak` (line 8 of Listing 5.22), the localization estimate is performed relying only on the range measurements from AN_0 and AN_3. If the associated circumferences intersect, the position estimate is chosen as the point with the smallest abscissa, otherwise the position estimate cannot be obtained. Finally, if $vPeak_3$ is below the threshold $vPeak_{\text{Th}}$ and it is the smallest value of `vPeak` (line 14 of Listing 5.22) the position estimate is performed using `findTargetPositionS0S2` defined in Listing 5.19.

In procedure `findTargetPositionNoS2VPeak` defined in Listing 5.23 the localization scheme is analogous to the one in Listing 5.21 and Listing 5.22, except for the fact that the unavailable or ignored range estimate is now the one relative to AN_2.

The procedure `findTargetPositionNoS3VPeak` in Listing 5.24 is similar to the previous ones, beside the fact that we discard the range estimate relative to AN_3.

Listing 5.23: Pseudocode of procedure `findTargetPositionNoS2VPeak`.

```
1  procedure findTargetPositionNoS2VPeak
2
3  if (vPeak_0 ≥ vPeak_Th ∧ vPeak_1 ≥ vPeak_Th ∧ vPeak_3 ≥ vPeak_Th)
4      findTargetPositionNoS2
5  else if (vPeak_0 < vPeak_1 ∧ vPeak_0 < vPeak_3)
6      I ← {p_0, p_1} = C_1 ∩ C_3
7      findTargetPositionS1S3
8  else if (vPeak_1 < vPeak_0 ∧ vPeak_1 < vPeak_3)
9      I ← {p_0, p_1} = C_0 ∩ C_3
10     if(I ≠ ∅)
11         u ← maxAbscissa(I)
12     else
13         u ← [∞, ∞]
14 else if (vPeak_3 < vPeak_0 ∧ vPeak_3 < vPeak_1)
15     I ← {p_0, p_1} = C_0 ∩ C_1
16     findTargetPositionS0S1
```

Finally, let us describe the procedure `findTargetPositionS2S3` used in Listings 5.21 and 5.22. This procedure is analogous to `findTargetPositionS0S1` defined in Listing 5.18, but, in this case, only $\mathscr{C}_2$ and $\mathscr{C}_3$ can be derived. If they intersect, the two obtained points are symmetric with respect to the line $\mathscr{R}_{23}$ which goes through their centers, whose equation is

$$\mathscr{R}_{23} : y = 0. \tag{5.19}$$

Since the range requests from AN_0 and AN_1 were not successful, it is reasonable to assume that the TN position can be estimated as the point further from them, namely the one with the smallest ordinate. If $\mathscr{C}_2$ and $\mathscr{C}_3$ do not intersect, we look for the two nearest points of them and if their distance is below the threshold d_{Th} the TN position is estimated as their baricenter. Otherwise, the position estimate cannot be performed.

Listing 5.24: Pseudocode of procedure `findTargetPositionNoS3VPeak`.

```
1  procedure findTargetPositionNoS3VPeak
2
3  if (vPeak_0 ≥ vPeak_Th ∧ vPeak_1 ≥ vPeak_Th ∧ vPeak_2 ≥ vPeak_Th)
4      findTargetPositionNoS3
5  else if (vPeak_0 < vPeak_1 ∧ vPeak_0 < vPeak_2)
6      I ← {p_0, p_1} = C_1 ∩ C_2
7      if(I ≠ ∅)
8          u ← minAbscissa(I)
9      else
10         u ← [∞, ∞]
11 else if (vPeak_1 < vPeak_0 ∧ vPeak_1 < vPeak_2)
12     I ← {p_0, p_1} = C_0 ∩ C_2
13     findTargetPositionS0S2
14 else if (vPeak_2 < vPeak_0 ∧ vPeak_2 < vPeak_1)
15     I ← {p_0, p_1} = C_0 ∩ C_1
16     findTargetPositionS0S1
```

Listing 5.25: Pseudocode of procedure `findTargetPositionS2S3`.

```
1  procedure findTargetPositionS2S3
2
3  if(I ≠ ∅)
4      u ← minOrdinate(I)
5  else
6      A ← C_2 ∩ R_23,  B ← C_3 ∩ R_23
7      I = {p_0, p_1} ← findNearer(A, B)
8      if (||p_0 − p_1|| ≤ d_Th)
9          u ← 1/2 (p_0 + p_1)
10     else
11         u ← [∞, ∞]
```

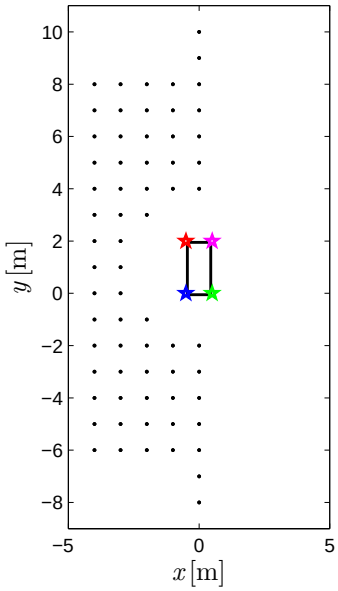


Figure 5.8: All the considered TN positions (*black dots*) are shown. The ANs (*coloured stars*) and the obstacle (*black rectangle*) are also shown.

5.5 Results Obtained with Four ANs

In this section, we show some results relative to the scenario described in Section 5.4. As in the case with three ANs in Section 5.3, the values of the parameters L and V in (5.3) are set to 2 m and 0.5 m, respectively, so that the coordinates of the ANs, expressed in meters, are

$$\underline{s}_0 = [-0.5, 2] \qquad \underline{s}_1 = [0.5, 2] \qquad \underline{s}_2 = [-0.5, 2] \qquad \underline{s}_3 = [-0.5, 2]. \tag{5.20}$$

All the results are obtained with the distance thresholds d_{Th} set to 0.5 m and the `vPeak` threshold $vPeak_{\mathrm{Th}}$ set to 5000 as in Section 5.3. We consider the same TN positions as in Section 5.3 in order to be able to compare the performances of the localization algorithms with three ANs, derived in Section 5.3, with those obtained here with four ANs. Such positions are shown in Figure 5.8, together with the positions of the four ANs and the metal obstacle among them.

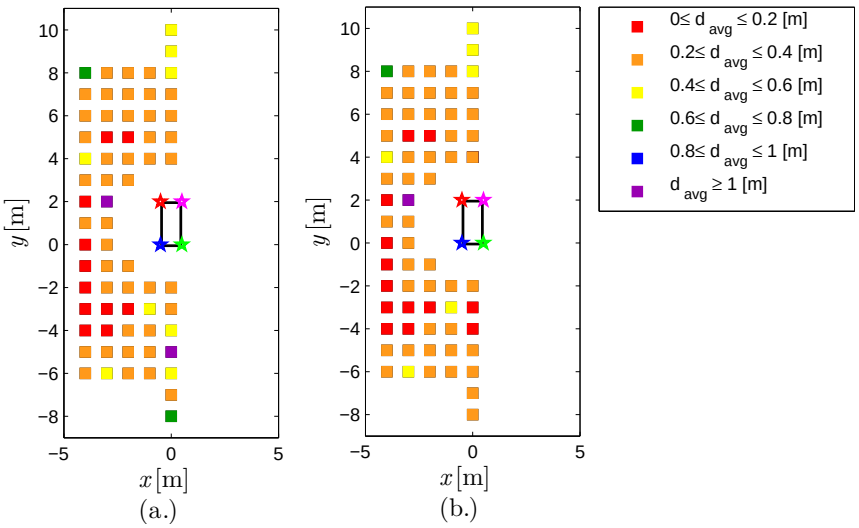


Figure 5.9: Values of d_{avg} relative to different TN positions for (a.) Algorithm 1 and (b.) Algorithm 2.

For each of the 66 TN positions in Figure 5.8 we apply Algorithm 1 described in Listing 5.9 with $N_c = 100$ and we evaluate the average distance error d_{avg} defined in (5.2), averaged over the 100 position estimates. Then, Algorithm 2 described in Subsection 5.4.2 is applied to the same range estimates used with Algorithm 1 and the average distance error d_{avg} is evaluated for each TN position.

The values of d_{avg} are shown in Figure 5.9. In particular, Figure 5.9 (a.) is relative to Algorithm 1. Each TN position is associated with a different colour, depending on the corresponding value of d_{avg}.

As in Figure 5.3: red squares correspond to values of d_{avg} smaller than 20 cm; orange squares correspond to values of d_{avg} between 20 cm and 40 cm; yellow squares correspond to values of d_{avg} between 40 cm and 60 cm; green squares correspond to values of d_{avg} between 60 cm and 80 cm; blue squares correspond to values of d_{avg} between 80 cm and 1 m; and violet squares correspond to values of d_{avg} greater than 1 m. Figure 5.3 (b.) is analogous to Figure 5.9 (a.) but it refers to the values of d_{avg} obtained with Algorithm 2.

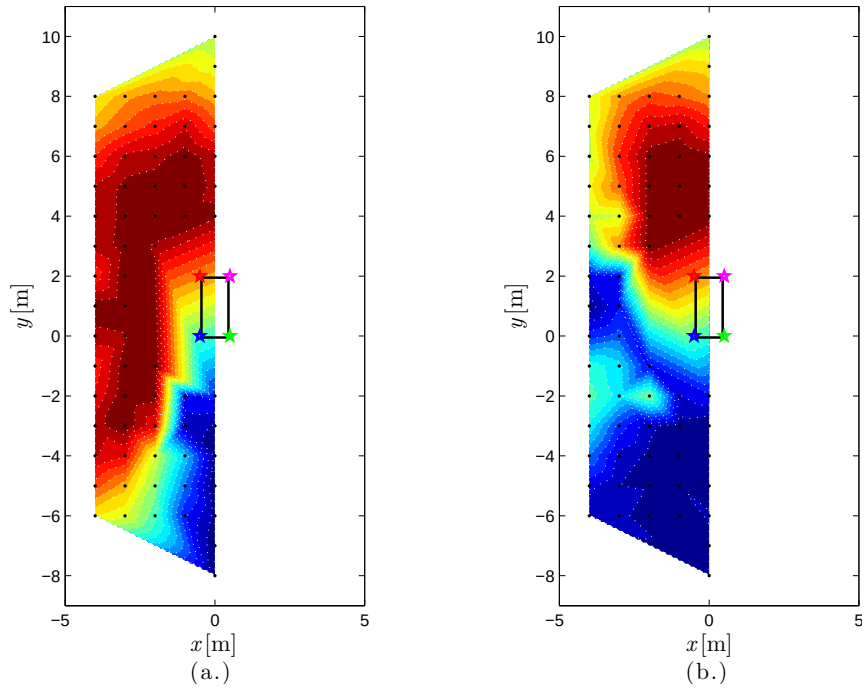


Figure 5.10: The values of (a.) $vPeak_0$ and (b.) $vPeak_1$ are shown.

In this case, the improvement brought from Algorithm 2 with respect to Algorithm 1 are less evident. As a matter of fact, Algorithm 2 improves the performance of Algorithm 1 only in 5 cases.

It is then of interest to compare the results in Figure 5.3, relative to the configuration with three ANs, with those in Figure 5.9. More precisely, a comparison between Figure 5.3 (a.) and Figure 5.9 (a.) shows that the use of an additional AN allows obtaining more accurate position estimates in various TN positions. As a matter of fact, in the scenario with four ANs, the values of d_{avg} are below 40 cm in 54 points, namely 81% of the times. This percentage in nearly twice the one relative to Figure 5.3 (a.). Moreover, the comparison between Figure 5.9 (a.) and Figure 5.3 (b.) shows that Algorithm 1 applied in the scenario with four ANs outperforms Algorithm 2 relative to the configuration with three ANs. As a matter of fact, in Figure 5.3 (b.) the values

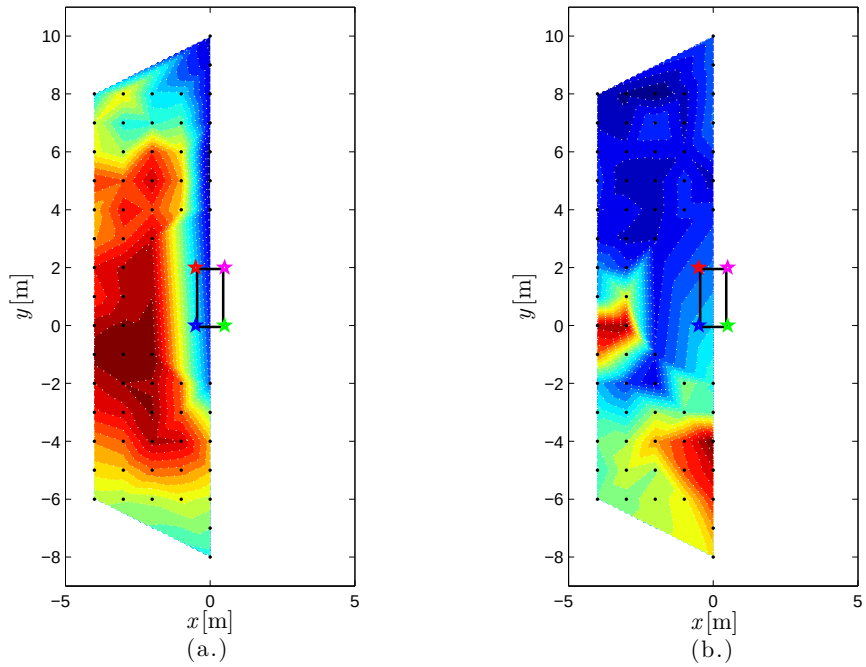


Figure 5.11: The values of (a.) $vPeak_2$ and (b.) $vPeak_3$ are shown.

of d_{avg} are below 40 cm only in 40 points, which is less than 20% less than in Figure 5.9 (a.). Finally, Figure 5.9 (b.) shows that Algorithm 2 applied in the scenario with four ANs allows keeping d_{avg} below 40 cm nearly 90% of the times. When applying Algorithm 2 to the case with three ANs, the same percentage is obtained if values of d_{avg} up to 80 cm are tolerated.

Finally, the values of $\{vPeak_i\}_{i=0}^{3}$ in the considered TN positions are shown in Figure 5.10 and Figure 5.11. More precisely, Figure 5.10 (a.) refers to $vPeak_0$ and Figure 5.10 (b.) refers to $vPeak_1$, while Figure 5.11 (a.) refers to $vPeak_2$ and Figure 5.11 (b.) refers to $vPeak_3$. Comparing these results with those in Figure 5.6 shows that four ANs guarantee a better coverage in all the considered TN positions, as confirmed by the results in Figure 5.9.

Conclusion

In this dissertation, the localization of targets in indoor environment by means of the Ultra Wide Band (UWB) technology has been investigated from analytical, simulative and experimental points of view.

In Chapter 1, the motivations of the importance of indoor localization are presented and an overview of the literature on this topic is given. A variety of approaches suitable to address this problem is described and the advantages brought by the UWB technology in this context are emphasized.

Aiming at finding strategies to improve the localization accuracy, the presented work initially focuses on optimized placement of Anchor Nodes (ANs) used for target localization. In particular, in Chapter 2, an analytical approach to optimized ANs placement for UWB-based localization of a target moving in a large indoor scenario is proposed. Imposing (realistic) constraints on the ANs positions and assuming that the target moves along a straight line in the middle of a corridor, a closed-form expression for the optimal distance between consecutive ANs is derived. The validity of the analytical framework is confirmed by simulations, which also show that the proposed placement strategy is effective even when the TN follows generic paths.

However, in practical application scenarios it is not always possible to decide the ANs positions. For this reason, to overcome the limitations of geometric techniques, which can suffer from ill-conditioning, an optimization-based approach to localization is proposed. More precisely, in Chapter 3, three approaches to UWB-based localization of nodes are considered: two stem from the geometry-oriented localization literature and one originates from the soft computing literature, namely

the Particle Swarm Optimization (PSO) algorithm. The obtained results show that the PSO approach guarantees, with respect to the other algorithms, a better accuracy in the position estimate. Furthermore, an improved version of the PSO algorithm is proposed. Simulation results show that the convergence speed of the improved algorithm is significantly higher than that of the PSO algorithm, leading to a relevant reduction of the total computational cost.

In order to perform a more realistic analysis of the localization accuracy, experimental results are then considered. In Chapter 4, a statistical model for the distance estimates in Line-of-Sight (LoS) conditions is derived on the basis of an experimental campaign of range measurements obtained using Time Domain PulsON 410 Ranging and Communications Modules (RCMs). The range estimate error between pairs of RCMs is statistically analyzed and the results show that its average and standard deviation are well approximated as linearly increasing functions of the true distance between the considered pair of nodes. The derived statistical model is applied to correct the predictions of two illustrative localization algorithms in two realistic localization scenarios. Results show that the distance corrections introduced by the proposed model allow improving the performance of the considered algorithms, thus confirming the validity of the model.

In Chapter 5, a real-world scenario is considered, where the presence of a metal obstacle introduces Non-Line-of-Sight (NLoS) effects making localization harder. First, two variants of a geometrically inspired algorithm for the localization of a target by means of three ANs are proposed. The results obtained in our experiments with PulsON 410 RCMs are shown for different target positions. Then, the adopted localization algorithm is rearranged for the case of four ANs. Results show that the use of an additional AN allows improving the performance of the localization in terms of the average distance between the true target positions and their estimates.

In conclusion, the possibility of using UWB technology for accurate indoor localization is investigated in this dissertation. The obtained results show that it is indeed possible to achieve 2-3 cm accuracy if all nodes are in LoS and 20-40 cm accuracy in NLoS scenarios.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 104–112, August 2002.

[2] J. M. Alonso, M. Ocana, N. Hernandez, F. Herranz, A. Llamazares, M. A. Sotelo, L. M. Bergasa, and L. Magdalena, "Enhanced WiFi localization system based on soft computing techniques to deal with small-scale variations in wireless sensors," *Applied Soft Computing*, vol. 11, no. 8, pp. 4677–4691, December 2011.

[3] S. Barrett, "Optimizing sensor placement for intruder detection with genetic algorithms," in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics*, New Brunswick, NJ, May 2007, pp. 185–188.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, October 2000.

[5] S. Busanelli and G. Ferrari, "Improved ultra wideband-based tracking of twin-receiver automated guided vehicles," *Journal of Integrated Computer-Aided Engineering*, vol. 19, no. 1, pp. 3–22, March 2012.

[6] R. Cardinali, L. D. Nardis, M. G. D. Benedetto, and P. Lombardo, "UWB ranging accuracy in high and low-data-rate applications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 4, pp. 1865–1875, April 2006.

[7] D. Cassioli, M. Z. Win, and A. F. Molisch, "A statistical model for the UWB indoor channel," in *IEEE Vehicular Technology Conference*, Rhodes, Greece, May 2001, pp. 1159–1163.

[8] Y. Chan and K. C. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905–1915, August 1994.

[9] R. Chávez-Santiago, I. Balasingham, and J. Bergsland, "Ultrawideband technology in medicine: A survey," *Journal of Electrical and Computer Engineering*, vol. 3, no. 2, pp. 1–9, 2012.

[10] A. Conti, D. Dardari, M. Guerra, L. Mucchi, and M. Z. Win, "Experimental characterization of diversity navigation," *IEEE Systems Journal*, vol. 8, no. 1, pp. 115–124, March 2014.

[11] D. Dardari, C. C. Chong, and M. Z. Win, "Threshold-based time-of-arrival estimators in UWB dense multipath channels," *IEEE Transactions on Communication*, vol. 56, no. 8, pp. 1366–1378, August 2008.

[12] D. Dardari and R. D'Errico, "Passive ultrawide bandwidth RFID," in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM '08), New Orleans, LA, December 2008, pp. 1–6.

[13] B. Denis, J. K. Keignart, and N. Daniele, "UWB measurements and propagation models for snowy environments," in *IEEE International Conference on Ultra-Wideband*, Zurich, Switzerland, September 2005, pp. 1–6.

[14] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber-physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, January 2012.

[15] R. Eberhart and J. Kennedy, "A new optimizer using particles swarm theory," in *Proceedings of the* 6^{th} *International Symposium on Micro Machine and Human Science* (MHS), Nagoya, Japan, October 1995, pp. 39–43.

[16] Z. Farid, R. Nordin, and M. Ismail, "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*, vol. 2013, 2013.

[17] W. H. Foy, "Position-location solutions by Taylor-series estimation," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 12, no. 2, pp. 187–194, March 1976.

[18] S. Gezici and H. V. Poor, "Position estimation via Ultra-Wide-Band signals," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 386–403, February 2009.

[19] S. Gezici, T. Zhi, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: A look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, July 2005.

[20] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 1, pp. 13–32, January 2009.

[21] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, October 2009.

[22] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proceedings of the* 9^{th} *Annual International Conference on Mobile Computing and Networking* (MobiCom), San Diego, CA, September 2003, pp. 81–95.

[23] K. C. Ho, X. Lu, and L. Kovavisaruch, "Source localization using TDOA and FDOA measurements in the presence of receiver location errors: Analysis and solution," *IEEE Transactions on Signal Processing*, vol. 55, no. 2, pp. 684–696, February 2007.

[24] K. C. Ho and W. Xu, "An accurate algebraic solution for moving source location using TDOA and FDOA measurements," *IEEE Transactions on Signal Processing*, vol. 52, no. 9, pp. 2453–2463, September 2004.

[25] N. Holden and A. A. Freitas, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," in *Proceedings of the IEEE Swarm Intelligence Symposium* (SIS), Pasadena, CA, June 2005, pp. 100–107.

[26] X. Huang, E. Dutkiewicz, R. Gandia, and D. Lowe, "Ultra-wideband technology for video surveillance sensor networks," in *IEEE International Conference on Industrial Informatics*, Singapore, August 2006, pp. 1012–1017.

[27] H. J. P. Iglesias, V. Barral, and C. J. Escudero, "Indoor person localization system through RSSI bluetooth fingerprinting," in *Proceedings of the* 19^{th} *International Conference on Systems, Signals and Image Processing*, Vienna, Austria, April 2012, pp. 40–43.

[28] M. B. Ismail, A. F. A. Boud, and W. N. W. Ibrahim, "Implementation of location determination in a Wireless Local Area Network (WLAN) environment," in *Proceedings of the* 10^{th} *International Conference on Advanced Communication Technology*, Hanoi, Vietnam, February 2008, pp. 894–899.

[29] J. Karedal, S. Wyne, P. Almers, F. Tufvesson, and A. F. Molisch, "UWB channel measurements in an industrial environment," in *IEEE Global Telecommunications Conference*, Dallas, TX, November 2004, pp. 3511–3516.

[30] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium* (SIS), Indianapolis, IN, April 2003, pp. 80–87.

[31] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks* (ICNN), Perth, Australia, November 1995, pp. 1942–1948.

[32] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the Conference on Systems, Man, and Cybernetics*, vol. 4, Orlando, FL, October 1997, pp. 4104–4109.

[33] A. Krause, J. Leskovec, C. Guestrin, J. Vanbriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 516–526, November 2008.

[34] S. Lee, K. C. Lee, M. H. Lee, and F. Harashima, "Integration of mobile vehicles for automated material handling using Profibus and IEEE 802.11 networks," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 3, pp. 693–701, June 2002.

[35] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1067–1080, November 2007.

[36] C. Mensing and S. Plass, "Positioning algorithms for cellular networks using TDOA," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP 2006), Toulouse, France, May 2006, pp. 513–516.

[37] S. A. Mitilineos, A. D. Nick, M. T. Effie, K. M. Dimitris, and S. C. A. Thomopoulos, "An indoor localization platform for ambient assisted living using UWB," in *Proceedings of the* 6^{th} *International Conference on Advances in Mobile Computing and Multimedia* (MoMM '08), Linz, Austria, November 2008, pp. 178–182.

[38] G. Molina and E. Alba, "Location discovery in wireless sensor networks using metaheuristics," *Applied Soft Computing*, vol. 11, no. 1, pp. 1223–1240, January 2011.

[39] A. F. Molisch, D. Cassioli, C. C. Chong, S. Emami, A. Fort, B. Kannan, J. Karedal, J. Kunisch, H. G. Schantz, K. Siwiak, and M. Z. Win, "A comprehensive standardized model for ultrawideband propagation channels," *IEEE Trans-*

actions on Antennas Propagation, vol. 54, no. 11, pp. 3151–3166, November 2006.

[40] S. Monica and G. Ferrari, "An analytical approach to optimized anchors placement in UWB-based indoor localization," in *Riunione annuale 2012 del Gruppo nazionale Telecomunicazioni e Teoria dell'Informazione* (GTTI), Villasimius, Italy, June 2012.

[41] S. Monica and G. Ferrari, "Auto-localization of static nodes in UWB wireless sensor networks," in *Riunione annuale 2013 del Gruppo nazionale Telecomunicazioni e Teoria dell'Informazione* (GTTI), Ancona, Italy, June 2013.

[42] S. Monica and G. Ferrari, "Hybrid swarm intelligent node localization in wireless sensor networks," in *XVI Portuguese Conference on Artificial Intelligence* (EPIA 2013), Angra do Heroísmo, Açores, Portugal, September 2013.

[43] S. Monica and G. Ferrari, "Impact of the number of beacons in PSO-based auto-localization in UWB networks," in *Proceedings of the International Conference on the Applications of Evolutionary Computation* (EvoApplications '13), *track on Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems* (EvoCOMNET '13), Vienna, Austria, April 2013, pp. 42–51.

[44] S. Monica and G. Ferrari, "Optimized anchors placement: An analytical approach in UWB-based TDOA localization," in *Proceedings of the* 9^{th} *International Wireless Communications & Mobile Computing Conference* (IWCMC 2013), Cagliari, Italy, July 2013, pp. 982–987.

[45] S. Monica and G. Ferrari, "Particle swarm optimization for auto-localization of nodes in wireless sensor networks," in *Proceedings of the* 11^{th} *International Conference on Adaptive and Natural Computing Algorithms* (ICANNGA '13), Lausanne, Switzerland, April 2013, pp. 456–465.

[46] S. Monica and G. Ferrari, "Accurate indoor localization with UWB wireless sensor networks," in *Proceedings of the* 23^{rd} *IEEE International Conference on*

Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2014), *track on Capacity-Driven Processes and Services for the Cyber-Physical Society* (CPS), Parma, Italy, June 2014, pp. 287–289.

[47] S. Monica and G. Ferrari, "Swarm intelligent approaches to auto-localization of nodes in static UWB networks," *Applied Soft Computing*, vol. 25, pp. 426–434, December 2014.

[48] S. Monica and G. Ferrari, "UWB-based localization in large indoor scenarios: Optimized placement of anchor nodes," *IEEE Transactions on Aerospace and Electronic Systems*, 2015, to appear.

[49] F. Mourad, H. Snoussi, M. Kieffer, and C. Richard, "Robust interval-based localization algorithms for mobile sensor networks," *International Journal on Distributed Sensor Networks*, vol. 2012, no. 1, pp. 1–7, August 2012.

[50] L. D. Nardis and M. G. D. Benedetto, "Positioning accuracy in Ultra Wide Band low data rate networks of uncoordinated terminals," in *Proceedings of IEEE International Conference on UWB*, Boston, MA, September 2006, pp. 611–616.

[51] D. Niculescu and B. Nath, "Ad hoc Positioning System (APS)," in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM '01), San Antonio, CA, November 2001, pp. 2926–2931.

[52] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, July 2005.

[53] T. Pavani, G. Costa, M. Mazzotti, A. Conti, and D. Dardari, "Experimental results on indoor localization techniques through wireless sensors network," in *Proceedings of the IEEE Vehicular Technology Conference*, Melbourne, Australia, May 2006, pp. 663–667.

[54] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence Journal*, vol. 1, no. 1, pp. 33–57, June 2007.

[55] T. A. A. Rahman, U. A. K. Chude-Okonkwo, R. Ngah, and S. Nunoo, "Time dispersion analysis for UWB channel in an outdoor environment," in *IEEE Asia Pacific Conference on Wireless and Mobile*, Bali, Indonesia, August 2014, pp. 109–112.

[56] J. Romme and B. Kull, "UWB opportunities for smart home applications," in *Proceedings of the IEEE International Symposium on Consumer Electronics*, Erfurt, Germany, September 2002.

[57] Z. Sahinoglu, S. Gezici, and I. Guvenc, *Ultra-wideband positioning systems: Theoretical limits, ranging algorithms and protocols*. Cambridge, U.K.: Cambridge University Press, 2008.

[58] A. Sani, A. Alomainy, G. Palikaras, Y. Nechayev, H. Yang, C. Parini, and P. S. Hall, "Experimental characterization of UWB on-body radio channel in indoor environment considering different antennas," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 1, pp. 238–241, January 2010.

[59] R. O. Schmidt, "A new approach to geometry of range difference location," *IEEE Transactions on Aerospace Electronics Systems*, vol. 8, no. 6, pp. 821–835, November 1972.

[60] G. Shen, R. Zetik, and R. S. Thomä, "Performance comparison of TOA and TDOA based location estimation algorithms in LOS environment," in *Proceedings of the* 5^{th} *Workshop on Positioning, Navigation and Communication* (WPNC'08), Hannover, Germany, March 2008, pp. 71–78.

[61] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation* (ICEC), Washington, DC, July 1999, pp. 69–73.

[62] F. Subhan, H. Hasbullah, A. Rozyyev, and S. T. Bakhsh, "Indoor positioning in bluetooth networks using fingerprinting and lateration approach," in *Proceedings of the International Conference on Information Science and Applications*, Jeju Island, Korea, April 2011, pp. 1–9.

[63] W. K. Tam and V. N. Tran, "Propagation modelling for indoor wireless communication," *Electronics and Communication Engineering Journal*, vol. 7, no. 5, pp. 221–228, October 1995.

[64] Time Domain, "PulsON 410 data sheet," June 2012, 320-0289D.

[65] Time Domain, "Ranging and communications application programming interface (API) specification version 2.1," June 2012, 320-0282E.

[66] Y. Tiyaworanan, S. Promwong, and P. Supanakoon, "Characterization of transmission loss with indoor and outdoor in UWB impulse radio systems," in *International Symposium on Communications and Information Technologies*, Bangkok, Thailand, October 2006, pp. 1129–1132.

[67] M. Vecchio, R. Lopez-Valcarce, and F. Marcelloni, "A two-objective evolutionary approach based on topological constraints for node localization in wireless sensor networks," *Applied Soft Computing*, vol. 12, no. 7, pp. 1891–1901, July 2012.

[68] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 2, pp. 102–124, May 2008.

[69] X. S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," *Communications in Computer and Information Science*, vol. 136, pp. 53–66, March 2011.

[70] S. E. Yoo, P. K. Chong, D. Kim, Y. Doh, M. L. Pham, E. Choi, and J. Huh, "Guaranteeing real-time services for industrial wireless sensor networks with IEEE 802.15.4," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3868–3876, June 2010.

[71] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, December 2009.

[72] J. Zhang, P. V. Orlik, Z. Sahinoglu, A. F. Molisch, and P. Kinney, "UWB systems for wireless sensor networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 313–331, February 2009.

[73] Y. Zheng, W. Chenshu, C. Tao, Z. Yiyang, G. Wei, and L. Yunhao, "Detecting outlier measurements based on graph rigidity for wireless sensor network localization," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 374–383, January 2013.