Contributions to Target Detection in Wireless Sensor Networks

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul
INGINERIE ELECTRONICĂ ŞI TELECOMUNICAŢII
de către

Ing. Ruxandra Ioana RUSNAC

Conducător ştiinţific:	Prof.univ.dr.ing Aurel- Ştefan GONTEAN
Referenţi ştiinţifici:	Prof.univ.dr.ing. Theodor PETRESCU
	Prof.univ.dr.ing. Mireca DĂBÂCAN
	Prof.univ.dr.ing. Alimpie IGNEA

Ziua susţinerii tezei: 28 septembrie 2012

Seriile Teze de doctorat ale UPT sunt:

1. Automatică	7. Inginerie Electronică şi Telecomunicaţii
2. Chimie	8. Inginerie Industrială
3. Energetică	9. Inginerie Mecanică
4. Ingineria Chimică	10. Ştiinţa Calculatoarelor
5. Inginerie Civilă	11. Ştiinţa şi Ingineria Materialelor
6. Inginerie Electrică	

Universitatea „Politehnica” din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Foreward

My choice for the study of wireless sensor networks in general, and localization algorithms in particular, resides in the expansion that these networks gained in everyday life in the past years. Even though in their early beginnings wireless sensor networks were used in the military area for the surveillance of mines in battlefields, the gradually expanded to civil applications, such as greenhouse monitoring or seismic and landslide monitoring and even in automotive applications.

The confirmation of my choice came along as I read more on this topic and mostly as I attended conferences and got in contact with researchers studying the same field. The most impressive piece of work I have seen in the field of localization was in Ghent, Belgium. The Interdisciplinary Institute for Broadband Technologies together with Tom-Tom (a company developing GPSs) developed an indoor localization system which they called hybrid. This system used localization algorithms together with indoor maps in order to produce an efficient localization solution.

The need to determine the position of the network nodes resides in the idea that it is useless to know that an event occurred if its position is unknown. This is true for all the applications above. A solid example of such an application is a greenhouse where it is necessary to modify the humidity only in a certain area. If the exact area where the measure is necessary is unknown the correction cannot be applied.

Localization algorithms for wireless sensor networks are challenging due to the many aspects that they have. One such aspect is the use of the received signal strength indicator to determine the position of nodes. This method is emerging in the problem of localization. Its pitfalls are all scenarios where reflections exist: indoor, outdoor urban etc. These scenarios determine multipath propagation and thus an attenuation of the signal measured at the receiver. This makes this more suitable for outdoor environments, as the experiments conducted for the thesis proved.

Another reason for studying localization using wireless sensor networks is their increasing presence in traffic control. Traffic control is implemented via magnetic sensor mounted in the roads. These magnetic sensors detect the presence of the vehicle when its magnetic field changes and then sends the information to an access point via radio waves. Such a traffic control mechanism can be implemented for freeway traffic control or for intersection traffic light control.

Another challenging issue is the error resulted from the algorithm. The smaller the error, the more accurate the algorithm is. The work that I have done for this thesis involves the study of several localization algorithms in simulation and in practice, as well as determining means to reduce the obtained errors so that the algorithms become more accurate.

Due to all the reasons presented here I believe that wireless sensor network localization is a topic for the future, as these networks can improve significantly the quality of our life

Timişoara, september 2012 Ruxandra Ioana RUSNAC

Familiei mele.

Rusnac, Ruxandra Ioana

Contributions to Target Detection in Wireless Sensor Networks

Teze de doctorat ale UPT, Seria 7, Nr. 48, Editura Politehnica, 2012, 130 pagini, 82 figuri, 59 tabele.

ISSN:1842-7014

ISBN:978-606-554-542-7

Cuvinte cheie:
wireless sensor networks, localization algorithms, positioning error, Xbee transceiver, received signal strength indicator (RSSI)

Rezumat,

The thesis is focused on wireless sensor networks node localization algorithms. The algorithms are studied from three points of view: state of the art, simulation and experimental. The state of the art presents an analitical survey of the latest wireless sensor network localization algorithms. The surveyed algorithms are simulated and then an experimental verification is conducted. Throughout these three steps the author’s contributions are highlighted. These contributions include original classifications, the development of a novel network topolgy as well as original simulations and result analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API	Application Programming Interface
ASIC(s)	Application Specific Integrated Circuit(s)
EEPROM	Electrically Erasable Programmable Read Only
FPGA(s)	Field Programmable Gate Array(s)
GPS	Global Positioning System
I2C	Inter Integrated Circuits
LPTD	Line Proxy Target Detection
MLE	Maximum likelihood estimation
MMDS	Modified multidimensional scaling
PC	Personal Computer
RAM	Random Access Memory
ROM	Read Only Memory
RSSI	Received Signal Strength Indicator
SDP	Semidefinite programming
SOCP	Second order cone programming
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
WCL	Weighted centroid localization
WSN(s)	Wireless sensor network(s)

1. WIRELESS SENSOR NETWORKS

1.1. Introduction to Wireless Sensor Networks

Wireless sensor networks are build up of a large number of spatially distributed sensors that have the purpose of monitoring physical and environmental conditions such as temperature, pressure, humidity etc. These networks were developed with the purpose of successfully replacing human activity/presence in remote, hostile and dangerous environments such as battlefields, jungles or oceans and seas. However, wireless sensor networks have made their way through to every day life and now they are used in such applications as healthcare, office applications, industrial process monitoring and so on.

Wireless sensor networks support a large number of applications. In what follows a few of these applications will be presented [Zur09]:

1. **Area monitoring**: the nodes are placed in an environment to be monitored. The motes collect information regarding a phenomenon going on in the environment: heat, pressure, sound etc.
2. **Greenhouse monitoring**: it is known that each species of plant has its own requirements regarding environmental air temperature and humidity. By means of wireless sensor networks these parameters can be efficiently monitored and corrected if necessary.
3. **Landslide monitoring**: the sensors are buried in the soil and monitor the vibrations of the land. Through such networks valuable information about landslides can be gathered, stored and processed.
4. **Earthquake monitoring**: similar to the landslide monitoring, the sensors detect movements of the crust of the earth. Based on the information gathered earthquake warning systems can be developed.
5. **Air monitoring**: the sensor network is used for pressure, temperature, humidity as well as air gas measurements.
6. **Animal monitoring**: wireless sensor networks are useful in monitoring animals in savannas, oceans etc. Wireless sensor networks are used in monitoring the migration of whales in oceans or in monitoring animals that lived in captivity, but that were recently released for integration in their natural habitat.
7. **Military applications**: the surveillance of battlefields was the main reason that lead to the appearance of these networks. The battlefields are monitored for mine presence or for any other tasks where human presence would be dangerous.

Automotive applications: wireless monitoring of tire pressure. This application is based on the presence of a pressure sensor in the tire. The information is sent by radio to the main computer in the car.

1.2. Wireless Sensor Networks for Location Detection

Even though the number of nodes is variable, the structure of a wireless sensor network for location detection is fixed and follows a pattern. The networks are organized in a decentralized and distributed manner at the same time. This feature is explained by the fact that each node in the network collects information and sometimes processes it and acts according to the results.

In a wireless sensor network for location detection the following elements can be identified [Hol05]:

- **Source** nodes are the network elements that provide information. The sources can be represented by sensor in the network or, sometimes, by actuators, that offer the feedback for a certain action.
- **Sinks** are the entities that require the information collected by the sources. The sink can be other nodes (sensors) in the network or they can be external devices such as PCs, PDAs etc.
- **The gateway** is the network element through which the information exists the network. The gateway can be easily be a sink or a PC. The gateway can represent the link between two networks or simply the means by which the information is forwarded for storage and processing to the management station.
- **The management station** works similar to a server. It is usually a PC that collects the information provided by the gateway. The information is usually stored and processed and, if necessary, redistributed into the network.

In a network the above elements are not singular. The networks will have a large and variable number of sources and sinks, as roles change in a network. There can be one or more gateways and management stations, as the different nodes can process different information.

The elements of a wireless network are not always static. Therefore the issue of mobility arises and it covers three aspects. The **first aspect** concerns the **mobility of the network nodes**. This type of mobility is determined by the application where the wireless network is used. It is possible for the entire network to move (as in whale migration) or only some of the nodes. Mobility requires the reorganization of the network. This should be done rapidly so that the network continues to function correctly.

The **second aspect** refers to **mobile sinks** that can exist in wireless sensor networks. The mobility of sinks gains importance when the sink is situated elsewhere than the information. This involves a fast network reorganization in the sense of determining the fastest and most efficient path through which the information should travel so that it reaches destination.

The **third aspect** is related to **target tracking applications**. These involve the mobility of the event to be monitored. This again requires network reorganization. Prior to the appearance of the target the network nodes are in an idle state. As the target travels through the network the nodes in its vicinity wake up and observe the event. When the target leaves their area the nodes go back to their idle state. An important aspect of event mobility resides in that nodes that are not in the vicinity of the target will not leave their idle state unless they are involved in the transfer of information to other areas of the network.

A typical wireless sensor network target tracking topology is presented by Wälchli et. al [Wal08]. This topology is known as the **mobile agent** paradigm and it respects the decentralized and distributed feature, as well as the event mobility feature presented above. The difference between the mobile agent paradigm and the event mobility resides in the roles that sensors are assigned. The mobile agent paradigm introduces a measurement-based leader election algorithm. This algorithm is centered on the target to be detected. Initially the nodes are in an idle state. The target starts to move through the network and the network nodes sense it. Next, a leader is elected from the group of sensors that detected the target. The rest of the nodes are in an idle or passive state. The purpose of the passive state is for nodes to function as warnings of the upcoming events for the rest of the network nodes. The leader can be regarded as a master and the master must have at least three slaves (the sensor nodes that were not elected as leaders). This can be regarded as a miniature network, which functions separately and at the same time in coordination, with the main network.

The newly-elected leader gathers the tracking and localization information from its slaves. The information is then transferred to the management station for storage and processing. By means of the gathered information, the leader estimates the distance to the event.

As the target move along the network, the masters and slaves change.

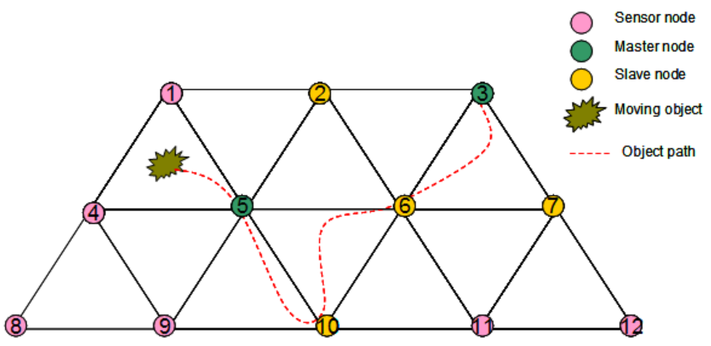


Figure 1.1 The mobile agent paradigm

1.3. Structure of a Wireless Sensor Network Node

Wireless sensor networks encompass a large variety of devices. Wireless sensor network is a collection of devices that have a processor and that possess sensing and communication capabilities. The sensor nodes are known as **motes**. A mote has a short range wireless communication, a small processor and a number of sensors. The mote is battery equipped and it is able to collect data from the environment where it is deployed, to communicate with other motes and to transfer information through the network [Hol05].

Motes must meet three requirements [Zur09]:

- Low manufacturing costs: the larger the network, the higher the cost
- Small size
- Energy-efficiency, as their power supply is a battery it should not discharge very fast

The structure of a mote is presented in Fig. 1.2

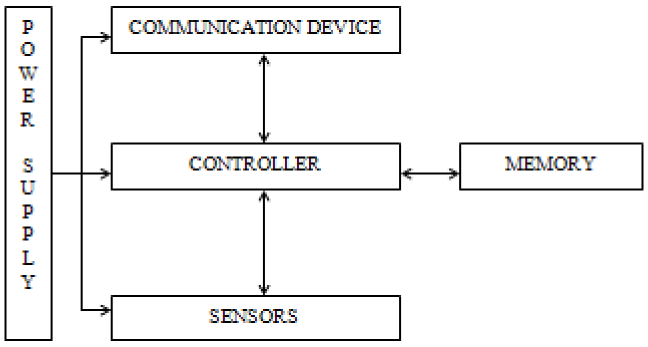


Figure 1.2 Structure of a sensor node (mote)

The central and most important element of the mote is the **controller** [Hol05]. Its role is to process and distribute data as well as to execute code. The controller is analogue to the central processing unit of a computer.

As a controller three devices can be used. One would be a general purpose processor. A drawback is the high energy consumption which is in contradiction with the energy-efficiency principle stated earlier.

The second device is the microcontroller. Microcontrollers have the great advantage of being used with embedded systems, of having a memory as well as low power consumption. Microcontrollers can enter the sleep mode (and reduce power consumption), which is of great importance for wireless networks.

Last but not least, Field Programmable Gate Arrays (FPGAs) and Application Specific Circuits (ASICs) are lately used as a controller. The advantage of ASICS is their flexibility as well as their reprogramability, but reprogramming comes with energy and time consumption. On the other hand, ASICs are custom-designed and well suited for high speed routers and switches. However, they loose flexibility to energy efficiency.

The **memory** [Hol05] stores software or any other temporary data. The type of memory to be used is chose depending on the application where the mote will be used. The Random Access Memory (RAM) is commonly used for storing sensor readings, transmission packets or any other data. However, RAM memories loose the stored information on power interrupt. The size of the memory is important when it comes to energy saving. However, no rules have been developed in order to establish the size of the memory and therefore the size is established according to the application where the mote is used.

There are other possibilities for the memory: Read Only Memory (ROM), Electrically Erasable Programmable Read Only Memory (EEPROM) or Flash. These types of memories are commonly used to store code. EEPROMs are used for the storage of nonvolatile data.

The **communication** [Hol05] is made up of two elements: the communication medium or channel and the hardware devices used for communication.

The communication medium can be optical, radio or ultrasound. For wireless networks the radio communication is the most frequent. This covers long distances and does not require the transmitter and the receiver to be in each other’s line of sight. It also provides high data rates as well as acceptable errors with low energy consumption. The frequency used in radio communication is typically between 433 MHz and 2.4 GHz.

The devices used for communication are called transceivers. They are data conversion devices in the sense that they convert a stream of bits from the microcontroller to and from radio waves. Transceivers make use of half duplex communication. In order to convert the data, transceivers incorporate devices like modulators, demodulators, amplifiers, mixers etc.

The **sensors** (and **actuators**) [Hol05] are the interface with the environment. They are the devices that observe and collect information from the environment. Sensors feature an area of coverage that defines a geographical surrounding form which the sensor can report accurate readings. In wireless sensor networks three types of sensors can be used:

- *Passive, onmidirectional sensors*: they collect the information from the environment without exploring it. This is the reason for which they are called passive. The onmidirectional feature is explained by the fact that the measurements made by these sensors are not affected by distance. Sensors that fall into this category are temperature sensors, light sensors, chemical sensors, humidity sensors etc.
- *Passive, narrow beam sensors*: the readings that these sensors make are affected by distances. For example a camera collects the information from a given direction, but if necessary, it can be rotated to change its line of sight.
- *Active sensors*: are the sensors that explore the environment. The sonar or the radar fall into this category.

The **power supply** [Hol05] determines the network's lifetime. Batteries are most frequently used as power supplies. Given the fact that wireless sensor networks are often placed in remote or dangerous places changing batteries is a challenging task. Another drawback of batteries is the limited lifetime that they have. This has an impact on the applications that can be sometimes loaded onto the node. In order to provide a longer lifetime for the motes **energy harvesting** is used.

Energy harvesting makes use of the resources available in the mote's environment in order to produce energy. The best candidates for energy harvesting are the renewable sources of energy, such as solar, wind, water or thermal, which are converted into the electrical power necessary for the motes to function. The greatest challenge that the conversion presents is the scale of the conversion device, which must be comparable to the size of the sensor node itself. The most frequently used renewable resources are solar, mechanical (vibrations) and thermal energy. [Sea09]

Solar energy is the most spread renewable resource in outdoor environments. Its disadvantage is that it offers enough power when sufficient light exists. Another problem is that the power obtained via solar light is not suitable for low power systems, such as a sensor node. This type of energy can be used in indoor environments as well, provided that the light system is functional at all moments and the system is operational at almost 100% duty cycle. For Crossbow/Berkley motes solar energy harvesting systems have been developed. This system is known as Helimote. [Sea09]

Mechanical energy is present almost in all places and take the form of vibrations. Vibrations, kinetic energy and mechanical energy generated by movements are forms of energy that can be harvested. Vibration energy can be harvested using piezoelectric sensors, while kinetic energy can be harvested using a spring-loaded mechanism. [Sea09]

The principle that lies under *thermal energy* harvesting is the current that is generated when there is a temperature difference between two junctions of a conducting material. Thermal energy harvesting uses temperature differences or gradients to generate electricity: between human body and the surrounding environment. Devices with direct contact to the human body can harvest the energy radiated from the human body by means of thermogenerators (TEGs). Due to the fact that thermal systems do not involve friction, such systems tend to have a longer lifetime that the mechanical energy harvesting systems. [Sea09]

The functioning of a sensor node is defined by two modes: the **initialization mode** and the **operation mode** [Hol05].

The initialization mode represents the stage in the node's functioning when it tries to integrate into the existing network. The node enters this mode when the power is turned on or when it detects a change in its environment. In this phase the node must detect its neighbors, determine its position and configure its hardware and software. All in all, the initialization mode is launched when the node establishes which services are available in the network and which services it should provide. It is the stage when the node basically determines the role it should play in the network's functioning.

The operation mode represents small bursts in node activity, such as sensor readings or data processing. The operation mode starts when the initialization mode ended and the node has reached a stable state known as the regular operation state. The parameters of the regular operation state are established during the initialization mode.

1.4. Wireless Sensor Network Modeling

Modeling wireless sensor networks is a complex issue. Due to the fact that most networks are created to serve a given application, the necessity to develop sensor or network models arises.

1.4.1. Wireless Sensor Node Models

The modeling of the sensor node depends very much on the type of application where the node is used, on the type of sensor and on many others. In literature a few models have been identified. A synthesis of these models is presented in Table 1.1. All the models listed are later discussed in detail.

From the sensor type point of view two models have been identified: the acoustic amplitude sensor model and the light sensor model.

The model for the *acoustic amplitude sensor* was developed by Feng and Guibas. An acoustic amplitude sensor measures the amplitude of a microphone and estimates the distance to the source using sound physics.

This approach regards the sound source as a point source and considers the propagation of sound as being lossless and isotropic. The root mean square amplitude measurement related to the sound position is expressed by (1):

Table 1.1 Wireless Sensor Network Node Models

Sensor model	Authors	Features
Threshold approach	Field & Grigoriu [Fie06]	• Considers measurement errors • Outcome classification
	Vall, Riley & Heck [Val06]	• Sensor reading is compared to a fixed threshold
	Banjeree et. al. [Ban08]	• Defines three categories of readings
Nearest sensor tracking model	Ling et. al. [Lin09]	• Based on received signal strength: the sensor with the highest signal reading tracks the target
Sensor type	Feng and Guibas [Fen04]	• Provides a model for acoustic sensors
	Wälchli et. al. [Wal08]	• Provides a model for light sensors

$$z = \frac{a}{\|x-\zeta\|} + w \tag{1}$$

In (1) *z* is the root mean square amplitude measurement, *x* is the position of the sound source, *a* is the root mean square amplitude of the sound source, *ζ* is the position of the sensor and *w* is a root mean square measurement noise. *w* is modeled as a Gaussian giving zero mean and the variance σ^2.

The model developed for the light source is similar to the model of the sound source. The light source can be modeled using (2):

$$\rho = \frac{c}{\|x-\zeta\|^{a}} + w \tag{2}$$

In (2) *ρ* is the received signal strength, c is the amplitude of the emitted signal, ζ *is the* position of the sensor node, *w* is the white Gaussian noise and *a* is the attenuation coefficient.

The value of the attenuation coefficient *a* depends on the environment where the network is placed. Table 1.2 presents the values of the attenuation coefficient for various environments [Zur09]:

Table 1.2. Pathloss coeffcient

Environment	Attenuation coefficient
Free space	2
Urban area cellular radio	2.7-3.5
Shadowed urban cellular radio	3.5-5
In building line of sight	1.6-1.8
Obstructed in building	4-6
Obstructed in factories	2-3

Wälchli et. al. used a value of a=2 for the modeling of the sound source. Equation (2) can be re-written in a nonlinear least squares form, which can be simulated:

$$f(x,c) = \sum_{i=1}^{k} (\rho_i - \frac{c}{\|x - \zeta_i\|^a})^2 \tag{3}$$

In (3) i=1…k represents the number of the network node. The network has a total of k nodes.

The difference between the acoustic amplitude sensor model and the *acoustic sensor* model is given by the fact that the *light sensor model* takes into account the environment in which it is placed by means of the attenuation coefficient. Despite this, the two models are very similar. For more accurate results the acoustic amplitude sensor model can be re-written to include the attenuation coefficient as well.

For the *threshold models*, three approaches have been identified. In the first approach, Field and Grigoriu propose a model that considers measurement errors. The considered sensor is a light sensor, similar to the one chosen by Wälchli et. al. The model proposes two vehicle classification states: g (good) and b (bad). The model aims at activating the sensor whenever a type b vehicle is detected. Taking into account the conditions presented above, the sensor output can be described as:

$$Y(t) = \sum_{k=1}^{N(t)} (Z_k + E_k) \quad (t = T_k),\ t \geq 0 \tag{4}$$

In equation (4) Z_k are the vehicle attributes, E_k is a collection of mean second order Gaussian random variables with the variance σ^2 and T_k are the jump times of $N(t)$. $N(t)$ depends on the vehicle arrival times at the sensor.

According to (4) the sensor classifies the vehicles using the following thresholds:

- If $Y(t) \geq \delta$ a type b vehicle passed at time t
- If $Y(t) < \delta$ no type b vehicle passed at time t

  $\delta \geq 0$ is a deterministic parameter for the sensor sensitivity.

A second threshold approach is presented by Vall, Riley and Heck. In this case the sensor reading is compared with a fixed threshold: a high sensor reading stands for an event, whereas low sensor readings indicate no event. The threshold is defined by equation (5):

$$th = \frac{m_e + m_n}{2} \tag{5}$$

In (5) m_e is the mean value of the sensor reading in the presence of an event and m_n is the mean value of the sensor reading in the absence of an event.

The last threshold approach is given by Banjeree et, al. The sensor readings are characterized by a sensor reading function F. This function must meet some properties:

- The senor reading of a given sensor must be independent from the other network sensors
- An upper and lower bound must exist for the normal readings' interval
- Within the normal readings' bound there exists a probability distribution function that be represented by means of a density function, *φ(i)*. The value of the density function is application specific, but in practice it can be approximated by a normal distribution function.

Based on the above conditions, Banjeree et. al. define three categories of readings. The first category is that of *normal readings* and it establishes that the value read by the sensor lies within the bounds that define normal readings and that the reading respects a normal distribution.

The second category of readings is called *event reading* and it occurs when the read value does not lie in the defined bounds. The *last category* is *called faulty reading* and it occurs when the outcome for normal and event readings are not clear and it is difficult to identify in which of the two categories lies the result. The event reading is defined by three subconditions that model the spatial correlation of sensor readings, the spatial properties of sensor readings, as well as the variations of irregular sensor readings.

The last model for the senor node is the one developed by Ling. et. al. This model is the nearest sensor tracking model and it is based on received signal strength. The model states that the sensor with the highest reading of the signal strength is responsible for tracking the target.

1.4.2. Wireless Sensor Network Models

This section is focused on some of the most common wireless sensor network models. A synthesis of the studies models is presented in Table 1.3, followed by a more detailed presentation.

Table 1.3 Wireless Sensor Network Models

Network Model	Authors	Features
Polygonal model	Tseng et. al. [Che03]	• Triangular cell network division • Mobile agents for roaming path tracking • Master-slave structure
	Lee et. al [Lee07]	• Sink-sensor target detection • Rectangular cell network division
Polygonal and graph model	Ling et. al. [Lin09]	• 2 submodels: network submodel and communication submodel
	Shih. et. al [Shi08]	• For heterogeneous networks

Ling. et. al. developed two submodels for the sensor network: a *sensing submodel* and a *communication submodel*. The sensor network is modeled as Voronoi graph. This approach divides the network in polygonal area around each sensor. Therefore when an object crosses the boundary of a graph a movement event will be called. The communication submodel constructs a communication graph on the basis of the sensors' transmission ranges.

Tseng. et. al. use a *polygonal model*. However, the network is divided into triangular cells. The algorithm uses mobile agents from the network in order to track the roaming path of the target. The assignment of roles in this case is similar to the approach of Wälchli et. al. As the target is detected the sensor nodes organize themselves in a master-slave structure. The mobile agent is the master. The target and the mobile agent move simultaneously; the movement of the mobile agent is determined by the target's trajectory. By means of the master-slave election mechanism, the triangular sensor areas around the moving object divide themselves into two categories:

- A working area, where three sensors that detect movement function normally
- Backup areas that delimit the areas where the mobile agent could move as the target moves. The backup areas and the working are direct neighbors.

Lee. et. al. present a *line proxy target detection* (LPTD). This algorithm uses the sink-sensor target detection mechanism, as well as a polygonal network division. The cells in the network are rectangular and are called clusters. For safety reasons, the data in a sensor node is stored in several nodes belonging to the same cell. The cells are labeled depending on their geographic position. However, all cell labels contain x and y in their name, as indices. All sensor nodes are aware to which cell they belong. The LPTD algorithm defines lines on the x axis and on the y axis by means of the cells belonging to the same row and column in the network. Each sink is delimited to the rectangular cell defined by certain rows and columns.

The algorithm establishes a method of time division. The divisions are called time durations and each lasts for δ_T. δ_T is known as line proxy rotation duration. During δ_T, the LPTD algorithm uses the cells in a line as proxies for sinks and targets. The algorithm assumes that sinks and sensors make use of the same temporal hash function. This makes the line act as a proxy at a certain moment of time.

The sensor networks presented above were comprised of the same type of sensors. Shih et. al. approach networks that make use of different type of sensors (*heterogeneous networks*). Despite this, the approach is similar to the ones presented above from the point of view of the graph approach, as well as from the point of view of the network division. The sensor network is modeled as an undirected simple graph. In order to characterize the events attributes are used. The attributes describing an event are given and each sensor node is aware of them. The area where multiple attribute regions overlap and all the attributes represent an event is named actual event region. The vertices of the graph are made up of sensor nodes of the same type. These vertices form a convex polygon called estimated attribute region

1.5. Wireless Sensor Network Node Localization

In a wireless sensors network it is necessary to know where an event has occurred in order to make sure that the proper corrections are applied at the correct place(s). In order to determine the positions of node in a network localization algorithms are used. These algorithms offer different approaches to the problem of localization and can be divided into two categories: **classical algorithms** and **modern algorithms**. This division was made according to the mathematical theory that lies behind the algorithms. *Classical algorithms* refer to algorithms that use direct calculus or such optimization methods as nonlinear least squares fitting or Newton-Raphson. The *modern algorithms* use convex optimization in order to perform calculations. Convex optimization is a recently developed method of optimization which has been used extensively in the past years. In what follows the two categories will be discussed.

1.5.1. Classical Algorithms

Classical algorithms are the maximum likelihood estimation algorithm, the weighted centroid localization algorithm, the modified multidimensional scaling, as well as the Malguki spring. Desai and Turelli [Des13] have conducted a simulation study concerning the maximum likelihood estimation, the modified multidimensional scaling as well as the weighted multidimensional scaling and the Malguki spring algorithm. Their work dealt with randomly deployed anchors and it studied the effect of the number of anchors over the obtained error. The authors tried to establish which algorithm provided the best results under identical conditions.

The modified multidimensional scaling was studied by K.W. Cheung and H.C. So [14]. The work contained a theoretical deduction of the algorithm as well as a numerical verification of the theoretical background.

An analysis of the *Malguki spring* was conducted by Arias et. al [Ari04]. The work contains a solid theoretical background as well as numerical results. The numerical analysis includes a comparison with other methods as well as an analysis of the effect of the number of anchors over the results.

A *modified version* of the *weighted centroid localization* called *selective adaptive weighted centroid localization* was developed by Fink and Beikirch [Fin11]. The algorithm improves of the accuracy by an adaptation of the weights according to their statistical distribution. The same authors study an algorithm called linear least squared error, which is in fact quite similar to the maximum likelihood estimation studied by Desai and Turelli [Des13]. The algorithms are studied at a practical level, as opposed to the previous approaches that offered a simulation study.

Except for Fink and Beikirch [Fin11], all the other authors study the algorithms in the situation of randomly placed anchors. Fink and Beikirch study the algorithms using anchors placed in line, at equal distances from each other. However, they only use two lines of anchors. None of the authors tried to offer a larger scale perspective over other anchor placement strategies, such as uniform anchors or other configurations that may results from these two.

Table 1.4 Classical algorithms computational time

Algorithm	**Computational Time [seconds]**
WCL	1.93
MLE	1.63
MMDS	0.88
Malguki	0.73

In Table 1.4 the computational times obtained by the author for the classical algorithms is presented. Theses values were obtained by averaging 100 samples.

1.5.2. Modern Algorithms

The modern approach to the problem of localization is concerned with convex optimization methods, such as second order cone programming and semidefinite programming. This approach has emerged recently and it is based on a more complex mathematical background than the classical algorithms.

Javanmard and Montari [Jan11] provide a theoretical approach to the problem of localization using *semidefinite programming*. What differs in this modern approach from the previous ones is the use of only distances between network elements. That is, the algorithm operates only with anchor-sensor distances as well as sensor-sensor distances. The positions of the anchors are not known and after applying the algorithm all the positions of the network elements are obtained. This algorithm is a three step method and the last two steps of the algorithm are identical to the last two steps used for the modified multidimensional scaling problem. These steps involve a best rank approximation of a computed matrix and then based on this approximation the results is obtained.

Shirazi et. al [Shi11]. study the localization of nodes using *second order cone programming*. This approach considers the problem from two points of view: perfectly known anchors positions as well as anchor position uncertainty (which is the case in most practical situations). The authors offer simulation results for the theoretical background as well as a performance analysis.

An analysis of localization with noisy (uncertain) distance data is done by Biswas et. al [Ban08]. The authors develop a *semidefinite model* for the problem of localization, model which they apply to noise affected distance information. The results are verified via simulation. What is different from the other approaches is the position of the anchors relative to the sensors. The authors consider inner anchors ("inside" the sensor positions) and outer anchors ("outside" the sensor positions).

In Table 1.5 the computational times obtained by the author for the modern algorithms are presented. As for the modern algorithms, for obtaining the computational times 100 samples were averaged.

Table 1.5 Modern algorithms computational time

Algorithm	**Computational Time [seconds]**
SOCP	2.60
SDP	1.68

1.6. Conclusions

The current chapter was meant as in introduction to wireless sensor networks in general and to localization in particular. In as far as the general introduction to wireless sensor networks is concerned general aspects as applications of the networks as well as the structure of a network node were presented. A careful analysis of the modeling possibilities was performed. The modeling can be done at network level and at node level. Different approaches encountered in literature were analyzed. A survey of wireless sensor networks for location detection was done. At this point the focus was on the mobile agent paradigm and the importance it has for this topic. The problem of localization could not be complete without a review of the most important node localization algorithms. This was accomplished via a classification of these algorithms in two branches, classical algorithms and modern algorithms.

Throughout this chapter it was shown that wireless sensor networks are very complex and their analysis comprises many aspects. A special focus was made on the problem of localization, which is the theme of the thesis.

1.7. Contributions

This chapter is a survey of the key topics in wireless sensor networks and therefore all the author's contributions to this chapter are theoretical contributions.One of the author's contributions is the comprehensive literary survey of wireless sensor networks from all the points of view discussed in this chapter.

The author introduces an original classification of the wireless sensor network modeling aspects, both from the point of view of the node as well as from the point of view of the whole network.

Another contribution to this is the original classification of wireless sensor networks presented in section 1.5. This classification divides the algorithms into classical and modern algorithms, depending on their mathematical background.

In sections 1.5 and 1.6 another contribution to this chapter can be found: the performance analysis of the presented algorithms given their computational time. The author obtained the presented values by means of original Matlab simulations.

1.8. References

1. [Zur09] Richard Zurawski, "Networked Embedded Systems", CRC Press, chapter 3, pp. 3-6, 2009
2. [Hol05] Holger Karl and Andreas Willig, "Protocols and Architectures for Wireless Sensor Networks" Wiley Interscience, pp. 17-67, 2005
3. [Wal08] Marcus Walchli, Samuel Bissig, Michael Meere and Torseten Braun, "Distributed Event Tracking and Classification in Wireless Sensor Networks", Journal of Internet Engineering, vol. 2, no.1,pp. 117-126, June 2008
4. [Fie06] R.V. Field Jr. and M. Grigoriu, "Optimal design of sensor networks for vehicle detection, classification and monitoring",Elsevier Probabilistic Engineering Mechanics., vol. 21, pp. 305-316, 2006
5. [Val06] ElMoustapha Ould-Ahmed-Vall and George F. Riley and Bonnie S. Heck, "A Distributed Fault-Tolerant Algoritm for Event Detection Using Heterogeneous Wireless Sensor Networks", Proceedings of 45th IEEE Conference on Decision and Control, USA, 2006.
6. [Ban08] Torsh Banerjee, Bin Xie and Dharma P. Agrawal, "Fault tolerant multiple event detection in a wireless sensor network", Elsevier J.ParallelDistrib.Comput, vol.68, pp. 1222-1234, 2008
7. [Lin09] Chih-Yu Ling, Yu-Chee Tseng and Yung-Chih Liu, "Imprecision-Tolerant Location Management for Object-Tracking in Wireless Sensor Network", British Computer Society The Computer Journal Advance Access, 2009 [Fen04] Zao Feng and Leonidas Guibas, "Wireless Sensor Networks-An Information Processing Approach", Elsevier Morgan Kaufmann Series in Networking, pp. 51-55, 2004
8. [Che03] Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee and Chi-Fu Huang, "Location Tracking in Wireless Sensor Newtorks by Mobile Agents and Its Data Fusion Strategies", The British Computer Society, The Computer Journal, vol. 47, no.4, pp. 448-460, 2003
9. [Lee07] Jangwon Lee, Wei Yu and Xonwen Fu, "Energy-efficient target detection in sensor networks using line proxies", International Journal of Communication Systems, Wiley Interscience, vol. 21, pp. 251-275, 2007
10. [Shi08] Kuei-Ping Shih, Sheng-Shih Wang, Huang-Chang Chen and Pao-Hwa Yang, "CollECT: Collaborative event detection and tracking in wireless

heterogeneous sensor networks", Elsevier Computer Communications, vol. 31, pp. 3124-3136, 2008

11. [Rus10] Ruxandra Ioana Rusnac, Aurel Gontean, "Modeling and simulation of wireless sensor networks for event detection", Proceedings of International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI),Timişoara, Romania, May 2010, pp. 535-540
12. [Des07] Jasmin Desai and Uf Turelli, "Evaluation Performance of Various Localization Algorithms in Wireless and Sensor Networks", 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio communications, 2007
13. [Che05] K.W. Chenung and H.C. So, "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements", IEEE Transactions on Signal Processing, Vol. 53, No.2, pp. 460-470, February 2005
14. [Ari04] Jagoba Arias et. al., "GPS-less location algorithm for wireless sensor networks", Elsevier Computer Communications, vol. 30, pp. 403-409, 2004
15. [Fin11] Andreas Fink and Helmut Beikirch., "Reliable Radio-based Human Tracking in Hazardous Environments", Wireless Congress- 2011 systems and applications, Munich, Germany, November 2011
16. [Jan11] Adel Javanmard and Andrea Montari, "Localization from Incomplete Noisy Distance Measurements", 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31 July-5 August 2011, pp. 1584-1588
17. [Rap02] Theodore S. Rappaport, "Wireless Communications- Principles and Practice", Prentice Hall Communication Engineering and Technologies Series, pp. 81-138, 2002
18. [Sea00] Seah, W.K.G. et. al., "Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP) - Survey and challenges", 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE, 2009, pp. 1-5, 2002
19. [Bal07] Baldwin,P et. al., "VISUALSENSE: Visual Modeling For WirelessSensor Networks and Sensor Network Systems", The Regents of The University of California, 2007
20. [Lin09] Lin, C-Y et. al., "Imprecision- Tolerant Location Management for Object-Tracking in Wireless Sensor Network", British Computer Society The Computer Journal Advance Access, 2009
21. [Shi11] Shirazi,Ghasem et. al. , "Second order cone programming for sensor network localization with anchor position uncertainty", 8th Workshop on Positioning Navigation and Communication (WPNC), pp. 51-55, 2011
22. [Jan07] Lee, J et al., "Energy-efficient target detection in sensor networks using line proxies", International Journal of Communication Systems, Wiley Interscience, vol. 21, 2007, pp. 251-275
23. [Yik08] Yick, J et, al. , "Wireless sensor network survey", Elsevier Computer Networks 52, 2008, pp. 2292-2330
24. [Aro04] Arora, A et, al. , "A line in the sand: a wireless sensor network for target detection, classification, and tracking", Elsevier Computer Networks 46, 2004, pp. 605-634

2. WIRELESS SENSOR NETWORKS NODE POSITIONING ALGORITHMS

Nodes in a wireless sensor network meant for localization are divided into two categories:

- **Anchors**: these are the network nodes with known and fixed positions
- **Sensors**: these are the network nodes with unknown positions (positions to be determined)

In order to determine the positions of nodes specific algorithms are used. These algorithms are based on two elements: the positions of anchor nodes and the distances between anchors and sensors. In this chapter, sections 2.1-2.analyze some of the most popular, more complex and more challenging localization algorithms encountered in literature.

The motivation that resides in the use of the two elements mentioned above resides in the energy efficiency principal that WSNs must respect, as well as in costs. It would be straightforward to equip all the network nodes with GPS. However this is an expensive solution, as networks might be composed of hundreds of nodes, it is not energy efficient as it requires additional hardware, and it most certainly will not work indoors or in the vicinity of tall buildings. Given all these aspects, a compromise solution was developed. This solution requires only a small number of network nodes to have GPS or to have preallocated positions. The rest of the nodes can determine their positions based on the anchors nodes and the pathloss model (or the received signal strength).

[Des07] study wireless sensor networks localization algorithms in a 16x16 meter space. The authors of [Des07] use 20 sensors (unknown position nodes) and 7 sensors, which are placed in different configurations. The network that [Des07] consider is a fully connected network. The algorithms are simulated for a transmission power of 0.5 dB and 1dB respectively. The results are presented by means of root mean square error. The errors obtained by [Des07] range between a minimum of 0.5 meters to a maximum of 5.5 meters, depending on the considered algorithm.

The pathloss model is based on the assumption that in vacuum a radio signal coming from a punctual source propagates circularly. The transmission of radio signals is adherent with the energy transport into the sender's neighborhood. This energy level flattens with the distance *d* to the sender, but it can be detected by a receiver.

The model has as a starting point Friis's equation. This equation computes the distance between a transmitter and a receiver using the transmission power P_{tx}, the received power P_{rx}, the antenna gains G_{tx} and G_{rx}, the system losses L and the wavelength λ_0 in ideal conditions: no reflection, no diffraction and no obstacles. Friis's equation is presented in (1) [Rap09]:

$$\frac{P_{rx}}{P_{tx}} = \frac{G_{rx}G_{tx}}{L}\left(\frac{\lambda_0}{4\pi d}\right)^2 \qquad (1)$$

Given a transmitter node T and a receiver node R, if T is able to detect the received power P_{rx}, the distance between T and R can be calculated by rearranging equation (1). By analyzing equation (1) it can be observed that the wavelength λ_0 and the distance d affect P_{rx} quadratically. The attenuation of a signal is defined as path loss (PL). The path loss is the logarithm of transmitting power to received power, in decibel [Rap09]:

$$PL(dB) = 10\log\frac{P_{tx}}{P_{rx}} = -10\log\left(\frac{G_{rx}G_{tx}}{L}\left(\frac{\lambda_0}{4\pi d}\right)^2\right) \qquad (2)$$

Based on the above, the mean power at the receiver situated at distance d from the receiver can be written as [Des07]:

$$P_{rx}(d) = \tilde{P}_r(d_0) - 10n\log\left(\frac{d}{d_0}\right) \qquad (3)$$

In (3), $\tilde{P}_r(d_0)$ is the mean received power at distance d_0. For indoor application d_0 is 1 meter and n is the pathloss coefficient. The pathloss coefficient is environmental-dependent and its values have been empirically determined and are listed in Table 2.1.
For free space (no obstacles present) $n = 2$ [Des07].

Table 2.1 Pathloss exponent values for various environments [Rap02]

Environment	**Pathloss exponent, *n***
Free space	2
Urban area cellular radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
In building line-of-sight	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factories	2 to 3

Further on [Des07]:

$$P_{rx}(d) = \tilde{P}_r(d_0) - 10n\log\left(\frac{d}{d_0}\right) + \varepsilon_{dB} \qquad (4)$$

In (4), ε_{dB} is a Gaussian distributed random variable having zero mean and the variance σ^2 [Des07].
The distance can be computed using [Des07]:

$$\tilde{d} = d_0 10^{\frac{-(P_{rx}(d)-\tilde{P}_{rx}(d_0))}{10n}} \quad (5)$$

The following sections contain the theoretical foundations of the algorithms studied in this thesis.

Except for the Malguki spring algorithm, all algorithms make use of the following notations:

- (x_0, y_0) is the position of the sensor node (node with unknown coordinates)
- (x_i, y_i) are the positions of the anchor nodes. There are n anchors
- d_i is the actual distance between the anchors and the sensor(s)
- $\tilde{d}_i$ is the error affected distance between the anchors and sensors
- ε_i is the measurement error that adds up to d_i in order to obtain $\tilde{d}_i$

2.1. Maximum Likelihood Estimation

The maximum likelihood estimation algorithm derives from triangulation and takes into account imperfect distance measurements represented by distance estimates, $\tilde{d}$ and the unknown error, □. Therefore, the distance equation becomes [Des07]:

$$\tilde{d} = d_i + \varepsilon_i \quad (6)$$

It is necessary to use more than three anchors and redundant distance in order to reduce errors. An overdetermined system of equations of the following type is obtained [Des07]:

$$\varepsilon_i = d_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \quad (7)$$

The ideal case is when □ becomes zero. Then the equation given in (7) becomes [Des07]:

$$-x_i^2 - y_i^2 + d_i^2 = (x_0^2 + y_0^2) - 2x_i x_0 - 2y_i y_0 \quad (8)$$

It is necessary to eliminate x_0^2 and y_0^2 . This is achieved by subtracting the last equation from the previous ones. There are n anchors in all. After subtracting, $n-1$ equations of the following form are obtained [Des07]:

$$-x_i^2 - y_i^2 + d_i^2 + x_n^2 + y_n^2 - d_n^2 = 2x_0(x_n - x_i) + 2y_0(y_n - y_i) \quad (9)$$

Using equations (10), (11) and (12) [Des07]:

$$X = 2\begin{bmatrix} (x_n - x_1) & (y_n - y_1) \\ (x_n - x_2) & (y_n - y_2) \\ \vdots & \vdots \\ (x_n - x_{n-1}) & (y_n - y_{n-1}) \end{bmatrix} \tag{10}$$

$$y = \begin{bmatrix} -x_1^2 - y_1^2 + d_1^2 + x_n^2 + y_n^2 - d_n^2 \\ -x_2^2 - y_2^2 + d_2^2 + x_n^2 + y_n^2 - d_n^2 \\ \vdots \\ -x_{n-1}^2 - y_{n-1}^2 + d_{n-1}^2 + x_n^2 + y_n^2 - d_n^2 \end{bmatrix} \tag{11}$$

$$b = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{12}$$

equation (9) can be re-written in the following manner [Des07]:

$$y = Xb \tag{13}$$

The solution to equation (13) is given by (14) [Des07]:

$$b = (X^T X)^{-1} X^T y \tag{14}$$

Maximum Likelihood Estimation is the most basic algorithm out of all the studied algorithms. It uses direct calculus for the position estimates.

2.2. Weighted Centroid Localization

The weighted centroid localization algorithm (WCL) uses the distances between anchors and sensors in the form of weights. The distances are calculated using received signal strength. After obtaining the distances, the position of each sensor node can be calculated using weighted centroid [Hol05]:

$$S_i(\tilde{x}_i ; \tilde{y}_i) = \frac{\sum_{j=1}^{n} (w_{ij} \cdot A_j)}{\sum_{j=1}^{n} w_{ij}} \tag{15}$$

In (15) w_{ij} is the weight between sensor i and anchor j and A_{ij} is the anchor position. The weight is represented by the inverse of the distance between the i^{th} sensor and j^{th} anchor [Hol05]:

$$w_{ij}(d_{ij}) = \frac{1}{d_{ij}^{\,g}} \tag{16}$$

In equation (16) g is called a degree. The role of the degree is to amplify short distances to anchors. Usually the best results (though unstable) are obtained when $g = 1$ [Hol05].

Weighted centroid localization is in appearance a simple algorithm due to the few calculations necessary. Its drawback is the use of optimization for the calculations, which can be time consuming depending on the number of iterations to be performed.

2.3. Iterative Trilateration

This approach makes use of at least three anchor nodes (x_i, y_i) and of the distances between the anchor nodes and the unknown nodes, d_i. The difference between the measured and estimated distance is calculated by means of (17) [Hol05]:

$$f_i(x_e, y_e) = | d_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} | \tag{17}$$

An initial value for the estimated position is necessary in order to start the algorithm.

Similar to the maximum likelihood estimation, iterative trilateration seeks to minimize the error between the real sensor position and the measured (calculated position). This minimization is done iteratively.

2.4. The Malguki Spring Algorithm

The Malguki spring algorithm represents an analogy to the mechanical science: each network node is regarded as a mass and all the masses are connected to each other through springs. However, each spring influences the node position by means of its force. The force of a spring is given by the distance between nodes.

Distances influence springs as follows: when the nodes are in their true positions the spring does not produce any force and the position does not need any corrections. Otherwise, when the nodes are not in their true position, the spring will produce a force and the necessary correction will be applied and the final position of the body minimizes the elastic energy of the entire system.

As stated in the introduction, the network is composed of nodes that have known positions and nodes that have unknown positions. The anchors' positions are denoted as $\vec{a}_i$, with $i = 1, \ldots, n$, where n is the total number of anchors. The position of the sensor is denoted by $\vec{s}$, while the distances between the anchors and the considered sensor are denoted by $\vec{d}_i$.

In mechanics, springs are characterized by relaxation distances. Similarly, in WSNs the relaxation distance is considered the distance between anchors and sensors, while the true position of the sensor will be at the relaxation distance of the spring.

The total force is expressed by equation (18):

$$\vec{m} = \sum_{i=1}^{n} \vec{m}_i = \sum_{i=1}^{N} \left(d_i - \left|\vec{s} - \vec{a}_i\right|\right) \cdot \frac{\vec{s} - \vec{a}_i}{\left|\vec{s} - \vec{a}_i\right|} \tag{18}$$

For each anchor in the network the total elastic force $\vec{m}$ must be calculated.

Based on equation (18) a location error can be defined for each of the network's anchors:

$$e_i = \left|\vec{d}_i - \left|\vec{s} - \vec{a}_i\right|\right|$$
$$e = \sum_{i=1}^{n} e_i \tag{19}$$

The algorithm aims at minimizing the total error e. In order to achieve this it is necessary to take into account the fact that $\vec{m}$ and the gradient of the error function from (19) have the same direction and opposite senses. Therefore it indicates the direction in which the estimation must be corrected in order to minimize the location errors. $\vec{m}$ is turned into a displacement by means of a conversion parameter γ. The conversion parameter is not constant throughout the calculations, but it changes with every iteration, so that the algorithm provides the best results. In order to update γ it is necessary to store all its values. In order to assure a certain stability for γ and auxiliary parameter called learning rate is introduced. The learning rate is denoted by $\gamma \in (0,1)$.

The Malguki spring algorithm requires an initial guess for the sensor position $\vec{s}$. The guess is then refined using the total elastic force $\vec{m}$ as follows :

A. Establish a starting value $\vec{s}$ for sensor, as well as in initial value for the conversion parameter γ.

B. For the k^{th} position the total elastic force $\vec{m}$ and the total error e are calculated.

C. The new location is calculated using γ_k :

$$\vec{r}_{k+1} = \vec{r}_k + \gamma_k \cdot \vec{m}_k \tag{20}$$

D. If $r_{k+1} \neq 0$ the current value of the total force is compared to with its previous value using equation (21):

$$\cos\theta = \frac{\vec{m}_k \cdot \vec{m}_{k-1}}{\left|\vec{m}_k\right| \cdot \left|\vec{m}_{k-1}\right|} \tag{21}$$

If the two forces share the same direction but have different senses the conversion parameter needs to be updated.

E. The overall error e is compared to a threshold. If this does not happen the solution to the algorithm is $\vec{r}_k$. Otherwise a new iteration is performed starting with step B. The complexity of Malguki spring is higher compared to the algorithms presented up to this point. Its drawbacks can be the number of operations performed as well as the necessity to use optimization in order to obtain the position estimate.

2.5. Modified Multidimensional Scaling

The MDS algorithm regards the dissimilarities between objects as distances and its purpose is to find coordinates to explain them. Classical MDS operates with noisy distance measurements between a set of points and by using these measurements it estimates the points' coordinates [Che05].

In addition to the anchor coordinates and the anchor-sensor distances, the algorithm makes use of the distances between two anchors. The distances are defined in equation (22) [Che05]:

$$d_{ij} = \sqrt{(x_i - x_j)^T (x_i - x_j)} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad i = 1,2,\ldots n,\ j = 1,2,\ldots,n \tag{22}$$

The classical MDS has a major drawback: in order to offer the correct solution the centroid of all the coordinates must be in the origin: $\sum_{i=0}^{n} x_i = [00]^T$, with n the number of anchor nodes and (x_i, y_i) being the sensor node(s) [Che05].

In order to overcome the above-mentioned drawback the algorithm was modified so that it works correctly for points that do not have the centroid in the origin.

The first step of the algorithm is building the dissimilarities matrix, D [Che05]:

$$D = \begin{bmatrix} 0 & r_1^2 & r_2^2 & \cdots & r_n^2 \\ r_1^2 & 0 & d_{12}^2 & \cdots & d_{1n}^2 \\ r_2^2 & d_{21}^2 & 0 & \cdots & d_{2n}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_n^2 & d_{n1}^2 & d_{n2}^2 & \cdots & 0 \end{bmatrix} \tag{23}$$

In D the following notations are made [Che05]:

- d_{ij} is the distance between the i^{th} and j^{th} anchor nodes: it is not affected by measurement errors as these points are well known
- r_i is the distance between anchor and sensor nodes: it is affected by measurement errors as these distances are generally measured through received signal strength indicator (RSSI) or time of arrival (TOA) techniques.

The obtained dissimilarities matrix is then double centered. Double centering is the process of subtracting the row and columns averages of a matrix from its elements and, in the end the average of all the matrix elements is added. This can be expressed in a more simplified manner as follows [Che05]:

$$B = -\frac{1}{2} J_{n+1} D J_{n+1} \tag{24}$$

With [Che05]:

$$Jn = I_{n+1} - \frac{1}{n+1} 1_{n+1} 1_{n+1}{}^{T} \tag{25}$$

In the above relation [Che05]:

- J_{n+1} is known as the double centering matrix
- I_{n+1} is an $(n+1)\times(n+1)$ identity matrix
- 1_{n+1} is an $(n+1)\times 1$ vector of ones.

The distances between anchor nodes and sensor nodes, as well as the distances between the anchors themselves can be written more explicitly [Che05]:

$$d_{ij}^{2} = x_i^T x_i + x_j^T x_j - 2x_i^T x_j \tag{26}$$

Equation (26) can be expressed as [Che05]:

$$\sum_{i=0}^{n} d_{ij}^{2} = \sum_{i=0}^{n} x_i^T x_i + (n+1) x_j^T x_j - 2\sum_{i=0}^{n} x_i^T x_j \tag{27}$$

By dividing both sides with $n+1$ and taking into account the fact that $\sum_{i=0}^{n} x_i = x_0$ the following is obtained [Che05]:

$$\frac{1}{n+1}\sum_{i=0}^{n} d_{ij}^{2} = \frac{1}{n+1}\sum_{i=0}^{n} x_i^T x_i + x_j^T x_j - \frac{2}{n+1} x_0^T x_j \,, \quad j = 0,1,\ldots,n \tag{28}$$

By replacing *i* with *j* the following is obtained [Che05]:

$$\frac{1}{n+1}\sum_{j=0}^{n} d_{ij}^{2} = x_i^T x_i + \frac{1}{n+1}\sum_{j=0}^{n} x_j^T x_j - \frac{2}{n+1} x_i^T x_0 \,, \quad i = 0,1,\ldots,n \tag{29}$$

By summing all the above mentioned equations and dividing them by $n+1$ the following result is obtained [Che05]:

$$\frac{1}{(n+1)^2}\sum_{i=0}^{i=0}\sum_{j=0}^{n} d_{ij}^2 = \frac{2}{n+1}\sum_{i=0}^{n} x_i^T x_i - \frac{2}{(n+1)^2} x_0^T x_0 \tag{30}$$

Substituting all the above equations the expression for $x_i^T x_j$ is obtained [Che05]:

$$\begin{aligned} x_i^T x_i = -\frac{1}{2}(d_{ij}^2 - \frac{1}{n+1}\sum_{i=0}^{n} d_{ij}^2 - \frac{1}{n+1}\sum_{j=0}^{n} d_{ij}^2) + \\ \frac{1}{(n+1)^2}\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}^2) + \frac{1}{n+1} x_0^T x_j + \frac{1}{n+1} x_i^T x_0 - \\ - \frac{1}{(n+1)^2} x_0^T x_0 \end{aligned} \tag{31}$$

The next step in the algorithm is decomposing B obtained in equation (24) in the following manner [Che05]:

$$B = \begin{bmatrix} B_{11} & b^T \\ b & B_1 \end{bmatrix} \tag{32}$$

In (32) the following notations were made [Che05]:

- B_{11} is the first diagonal element of matrix B in equation (24)
- b is an $n \times 1$ column vector
- B_1 is an $n \times n$ matrix.

Similarly to B a modified scalar product matrix B' will be constructed. B' can be decomposed similarly to B [Che05]:

$$B' = \begin{bmatrix} B'_{11} & b'^T \\ b' & B'_1 \end{bmatrix} \tag{33}$$

From equation (33) the following elements need to be calculated: B'_{11}, b' and B'_1.

However [Che05]:

$$B'_1 = X_{BS} X_{BS}^T \tag{34}$$

where X_{BS} represents the distances between anchor nodes, which are not affected by errors [Che05].

$$X_{BS} = [x_1 x_2 \dots x_M]^T \tag{35}$$

Starting from (31) the expressions of $x_i^T x_0$, $x_0^T x_j$ and $x_0^T x_0$ can be obtained [Che05]:

$$x_i^T x_0 = -\frac{1}{2}\left(\frac{n+1}{n}\right) \times (d_i^2 - \frac{1}{n+1} \sum_{j=0}^{n} d_{ij}^2 + \frac{1}{(n+1)^2} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij}^2) + \frac{1}{n+1} x_0^T x_0 \tag{36}$$
$$i = 1,2,\dots,n$$

,

$$x_0^T x_j = -\frac{1}{2}\left(\frac{n+1}{n}\right) \times (d_j^2 - \frac{1}{n+1} \sum_{i=0}^{n} d_{ij}^2 - \frac{1}{n+1} \sum_{j=0}^{n} d_j^2 + \frac{1}{(n+1)^2} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij}^2) + \frac{1}{n+1} x_0^T x_0 \tag{37}$$
$$j = 1,2,\dots,n$$

$$x_0^T x_0 = -\frac{1}{2}\left(\frac{n+1}{n}\right)^2 (-\frac{2}{n+1} \sum_{i=0}^{n} d_i^2 + + \frac{1}{(n+1)^2} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij}^2) \tag{38}$$

From the above B' can be obtained [Che05]:

$$B' = \begin{bmatrix} (\frac{n+1}{n})B_{11} & (\frac{n+1}{n})b^T + (\frac{n+1}{n^2})B_{11}1_n^T \\ (\frac{n+1}{n})b + (\frac{n+1}{n^2})B_{11}1_n & X_{BS}X_{BS}^T \end{bmatrix} \tag{39}$$

In what follows the steps of the MMDS algorithm are summarized [Che05]:

1. Decompose B and thus obtain B_{11} and b; calculate B_1'
2. Obtain B'
3. Decompose B' using eigenvalues and eigenvectors as $B' = V\lambda V^T$, where $\lambda = diag(\lambda_1, \dots, \lambda_{n+1})$ and $\lambda_1 \ge \lambda_2 \ge \dots \ge \lambda_{n+1} \ge 0$; $V = [v_1, \dots, v_{n+1}]$. A possibility of having negative eigenvalues exists. However, the eigenvalues used here for calculations must be all positive

and thus it is necessary to ignore all the negative eigenvalues.

4. The principle axes solution is calculated. This is done by selecting the greatest first two eigenvalues and their eigenvectors: $\hat{X}^r = V_2 \lambda_2^{1/2}$, where $\lambda_2^{1/2} = diag(\lambda_1^{1/2} \lambda_2^{1/2})$ and $V_2 = [v_1 v_2]$

5. Partition $\hat{X}^r$ as $\hat{X}^r = \left[\hat{x}_0^r \hat{x}_{BS}^{rT}\right]^T$, where $\hat{x}_0^r$ and $\hat{x}_{BS}^{rT}$ are the rotated coordinates of the MS and BS respectively. The next step is to determine the requisite rotation matrix when $\sum_{i=1}^{n} x_i = [00]^T$:

$$\Omega' = \left(\hat{X}_{BS}^{rT} X_{BS} X_{BS}^T \hat{X}_{BS}^r \right)^{\frac{1}{2}} \left(X_{BS}^T \hat{X}_{BS}^r \right)^{-1} \tag{40}$$

6. Perform a good translation so that the sensor coordinates are obtained on the basis of the known anchor coordinates.

Even though the algorithm is based on direct calculus (matrix operations), the large number of steps required for the position estimate can be time consuming and can represent a disadvantage of this algorithm.

2.6. Comparative Analysis of the Localization Algorithms

In Table 2.2 a comparative analysis of the algorithms surveyed in sections 2.1-2.5 is presented.

Table 2.2 Algorithm Analysis

Algorithm	Complexity	Uses optimization	Computational time
WCL	Medium	Yes	1.93
MLE	Low	No	1.63
MMDS	high	yes	0.88
Malguki spring	High	Yes	0.73

For the comparative analysis three criterions are used: the algorithm complexity, the use of optimization for calculus and the computational time. These three features influence the hardware structure of the node: the need for additional hardware, the microcontroller size as well as the physical dimensions of the node. Last but not least, the speed at which these computations are performed may be essential to the applications where the nodes are used. For example, in a mine field surveillance application the algorithm should be fast (and extremely accurate) so that the warnings are sent in useful time. For monitoring a greenhouse the speed of the algorithm is not very important, as no life depends on the algorithm feedback.

Considering all the aspects above, an analysis of Table 2.2 can be performed. The complexity of the algorithms reveals the existence of one low complexity algorithm: the maximum likelihood estimation. Low complexity involves few calculations due to the fact that MLE does not use optimization. The WCL requires optimization for calculus, but the algorithms involves few steps and thus the complexity is medium and the computational time is average. The slowest algorithms are Malguki spring and MMDS. These algorithms require optimization and a large number of steps to reach the result. Therefore the algorithms require high computational time (they are quite slow).

An important aspect to consider is the size of the network. If the network is large, the algorithms that require optimization will require even more computational time than for a small network.

The analysis of the five surveyed algorithms pointed out the advantages and the disadvantages of each algorithm. The conclusion is, that when making a choice for one of the algorithms a tradeoff has to be made between complexity, speed, the hardware design aspects and the application where the algorithm will be used. Another aspect that needs to be considered is the continuous evolution of microcontrollers: they are becoming more energy efficient without loosing speed. Keeping all these aspects in mind the appropriate choice for one or more of the algorithms can be made. A last aspect to be considered is the error provided by each algorithm. This aspect will be discussed in the following chapters.

Table 2.3 presents a synthesis of the results ontained by other authors in similar conditions to the ones used by myself. These results are presented in the form of a positioning error, for the algorithms considered in the previous chapters.

Table 2.3 Survey of results from literature

Author	Algorithm	No. of anchors	Positioning error [meters]
[Des07]	MLE	3	5.2
		5	5.5
[Des07]	Malguki spring	3	4.1
		5	3.5
[Blu07]	WCL	4	2.5-5.3
[Des07]	MMDS	3	4.5
		5	3.5
[Ari07]		4	4.1
		10	3

The results in Table 2.3 are the reference for the author’s results. The author aims to obtain smaller positioning errors than the ones listed in Table 2.3.

2.7. Conclusions

In the current chapter the author presented an analytical survey of some of the most popular, more complex and more challenging localization algorithms encountered in literature. The algorithms considered for analysis were the maximum likelihood estimation, the weighted centroid localization, the iterative trilateration, Malguki spring and modified multidimensional scaling. For each algorithm the steps necessary for implementation were presented. In section 2.6 a comparative analysis of the algorithms was performed by the author. This analysis was introduced by means of a table and it considered such aspects as the algorithm complexity, if the algorithm uses optimization or not and the algorithm computational time. Such an analysis is very important before making a choice for an algorithm as it involves safety aspects, node design aspects etc. Another aspect to consider is the network size, which influences the speed and type of the algorithm to use.

2.8. Contributions

The author's contributions to this chapter are as follows

A contribution is the analysis of the localization algorithms. The algorithms are presented from a theoretical point of view and the most important aspects of each of the considered algorithms are highlighted.

Another contribution is the original comparative analysis of the considered algorithms. This analysis is made by means of Table 2.2. Here three aspects are considered: the algorithm complexity, if the algorithm uses optimization pr not and the computational time. When performing the analysis the author highlights the tradeoff that must be made between computational time and algorithm complexity when thinking of real life applications.

Obtaining the computational times of each algorithm is done by the author by means of original Matlab simulations.

2.9. References

1. [Des07] Jasmin Desai and Uf Turelli, "Evaluation Performance of Various Localization Algorithms in Wireless and Sensor Networks", 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio communications, 2007
2. [Che05] K.W. Chenung and H.C. So, "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements", IEEE Transactions on Signal Processing, Vol. 53, No.2, pp. 460-470, February 2005
3. [[Ari04] Jagoba Arias et. al., "GPS-less location algorithm for wireless sensor networks", Elsevier Computer Communications, vol. 30, pp. 403-409, 2004
4. [[Fin11] Andreas Fink and Helmut Beikirch., "Reliable Radio-based Human Tracking in Hazardous Environments", Wireless Congress- 2011 systems and applications, Munich, Germany, November 2011
5. [Jan11] Adel Javanmard and Andrea Montari, "Localization from Incomplete Noisy Distance Measurements", 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31 July-5 August 2011, pp. 1584-1588
6. [Rap02] Theodore S. Rappaport, "Wireless Communications- Principles and Practice", Prentice Hall Communication Engineering and Technologies Series, pp. 81-138, 2002
7. [Hol05] Holger, Karl and Andreas, Willig, "Protocols and Architectures for Wireless Sensor Networks", Wiley Interscience, pp. 60-62, 111-146, 2005
8. [Rus10_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Maximum Likelihood Estimation Algorithm Evaluation for Wireless Sensor Networks", Proceedings of 12th International Symposium on Symbolic and Numeric algorithms for Scientific Computimg (SYNASC),Timişoara, Sept. 2010, pp. 95-98

9. [Rus10_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Target detection algorithm validation in WSN", 9th International Symposium on Electronics and Telecommunications (ISETC), Timişoara, Nov. 2010, pp. 373-376
10. [Rus11_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Evaluation of wireless sensor networks localization algorithms", Proceedings of 6th International Conference Intelligent Data Acquisition and Advanced Computing (IDAACS),Prague, Sept. 2011, pp. 857-862
11. [Rus11_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Evaluation of some node localization algorithms", SIITME 2011, Oct. 2011, pp. 287-290
12. [Rus11_03] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Performance analysis of wireless sensor network node localization algorithm", SCVT 2011, Nov. 2011, pp. 1-5
13. [Lee07]Jangwon Lee, Wei Yu and Xonwen Fu, "Energy-efficient target detection in sensor networks using line proxies", International Journal of Communication Systems, Wiley Interscience, vol. 21, pp. 251-275, 2007
14. [Shi08] Kuei-Ping Shih, Sheng-Shih Wang, Huang-Chang Chen and Pao-Hwa Yang, "CollECT: Collaborative event detection and tracking in wireless heterogeneous sensor networks", Elsevier Computer Communications, vol. 31, pp. 3124-3136, 2008
15. [Rus10] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Modeling and simulation of wireless sensor networks for event detection", Proceedings of International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI),Timişoara, Romania, May 2010, pp. 535-540
16. [Des07] Jasmin Desai and Uf Turelli, "Evaluation Performance of Various Localization Algorithms in Wireless and Sensor Networks", 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio communications, 2007
17. [Che05] K.W. Chenung and H.C. So, "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements", IEEE Transactions on Signal Processing, Vol. 53, No.2, pp. 460-470, February 2005
18. [Ari04] Jagoba Arias et. al., "GPS-less location algorithm for wireless sensor networks", Elsevier Computer Communications, vol. 30, pp. 403-409, 2004
19. [Fin11] Andreas Fink and Helmut Beikirch., "Reliable Radio-based Human Tracking in Hazardous Environments", Wireless Congress- 2011 systems and applications, Munich, Germany, November 2011
20. [Jan11] Adel Javanmard and Andrea Montari, "Localization from Incomplete Noisy Distance Measurements", 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31 July-5 August 2011, pp. 1584-1588
21. [Rap02] Theodore S. Rappaport, "Wireless Communications- Principles and Practice", Prentice Hall Communication Engineering and Technologies Series, pp. 81-138, 2002
22. [Sea00] Seah, W.K.G. et. al., "Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP) - Survey and challenges", 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE, 2009, pp. 1-5, 2002

3. ALGORITHM IMPLEMENTATION AND VALIDATION

The current chapter introduces original simulation results of the five algorithms presented in the previous chapter: the maximum likelihood estimation algorithm, the weighted centroid localization algorithm, the iterative trilateration, the Malguki spring and the modified multidimensional scaling.

3.1. Simulation Conditions

The simulations for each of the presented algorithms were performed by the author using Matlab, for the following scenarios:

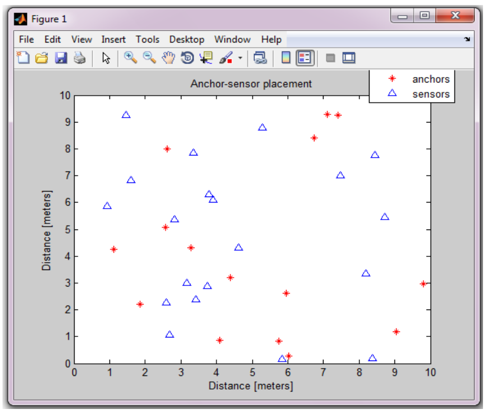


Figure 3.1a Random anchors

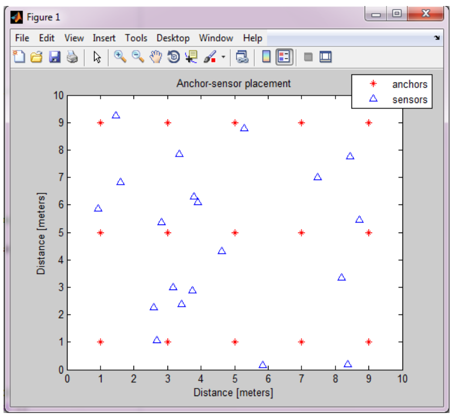


Figure 3.1.b Unifom anchors

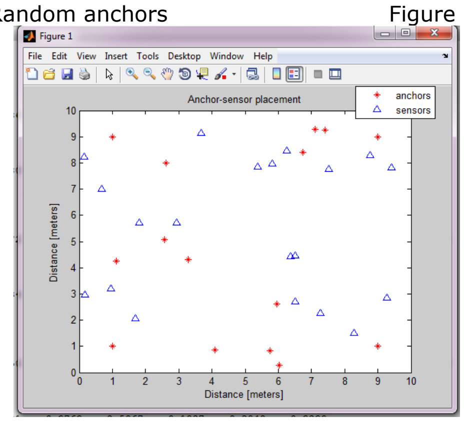


Figure 3.1.c Hybrid anchors

The scenario consists of a 10x10 meter room where 20 sensors are deployed. The number of anchors is varied between a minimum of 3 and a maximum of 15. The obtained network is fully connected.

Three anchor configurations were used: random, uniform and hybrid. The scenarios are illustrated in Fig. 3.1a, 3.1b and 3.1c.

The random anchors are the anchors that are randomly deployed in the considered space.

The uniform anchors are equally spaced from each other and from the corners of the room. This configuration resembles a uniform geometrical grid.

The **original hybrid configuration** was developed after observing the errors obtained from the previous two configurations. The hybrid anchors are meant to come as an error reduction mechanism, especially for the random placement. The hybrid deployment consists of 4 anchors, each placed 1 meter from the corners of the room, while the rest of the anchors are randomly deployed in the room. This can be regarded as a semirandom anchor placement method and it aims to assure a better coverage of the monitored surface.

For simulation purposes 10 different sensor configurations were analyzed while keeping the anchor position identical. As the number of sensors was varied between 3 and 15 an average error was calculated each time. The positioning error takes the form of a Euclidian distance, as expressed in equation (41):

$$err = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \quad (1)$$

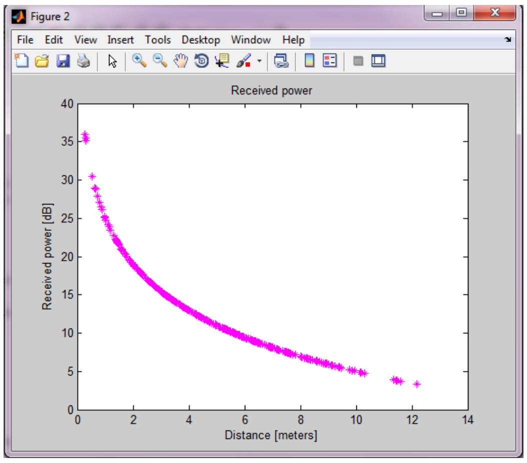


Figure 3.2 The pathloss model

The network considered here is fully connected. This means that all the networks nodes are within each other's range of communication.

At the beginning of the simulation the path loss model is simulated and verified. The graph expressing the decrease of the received power with distance is presented in Fig. 3.2. The simulations were performed considering the contribution of the earth reflected waves differs less than 5 dB form the pathlooss model.

3.2. Maximum Likelihood Estimation (MLE)

Table 3.1 contains a selection of the simulation results for the MLE.

Table 3.1 MLE positioning errors

No. of anchors	Random anchors	Uniform anchors	Hybrid anchors
3	3.33 m	1.44 m	1.13 m
5	2.10 m	1.01 m	0.80 m
7	1.30 m	0.89 m	0.83 m
10	1.02 m	0.83 m	0.80 m
13	0.64 m	0.78 m	0.69 m
15	0.64 m	0.72 m	0.58 m

A first analysis of Table 3.1 reveals that the highest errors are obtained for the use of random anchors. As expected, the error decreases as the number of anchors increases.

The random anchor placement yields an error of 0.64 meters for 15 anchors, while for 3 anchors the average error is 3.33 meters. An error of 0.64 meters is acceptable; however, errors of over 3 meters are quite high.

The next anchor set-up that was simulated was the uniform placement. The uniform placement provides an error of 1.44 meters for 3 anchors, while for 15 anchors the error is 0.72 meters. For the minimum number of used anchors the error reduces to more than half when compared to the random placement. When using 15 anchors the errors are comparable.

Still, the need to determine a means of reducing the error for random anchors persists. In this context it is necessary to maintain a certain degree of randomness. In order to achieve this, the orignal hybrid configuration was tested. The results obtained for small numbers of anchors are an improvement when compared to both the random and uniform placement. For large numbers of anchors the error is reduced as well. Therefore the goal of reducing the errors for the random placement was achieved and, more over, the errors introduced by the uniform placement were reduced as well.

A graphical illustration of the actual node position and the simulated node position is presented in Figs. 3.3a- 3.3f. The simulated and the real sensor positions are linked by a straight line. It is easy to notice that for small numbers of anchors the simulated and the real positions are farther apart and accordingly the lines uniting the points are longer. In opposition, the more anchors used the closer the points are.

In Fig. 3.4 is a comparative chart of the evolution of the obtained average errors for the 3 anchor configurations. By analyzing the chart a first observation can be made: the average error decreases as the number of anchors increases. The largest average error are obtained for 3 anchors, while the smallest average errors are obtained for 15 anchors, independent of the anchor placement scenario.

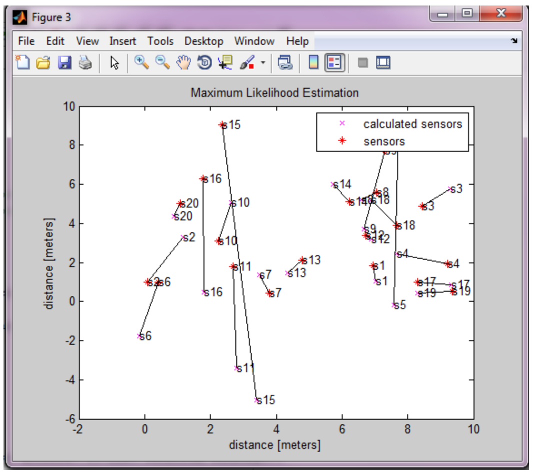


Figure 3.3a MLE, random anchors: real vs. simulated node position for 3 anchors and 20 sensors

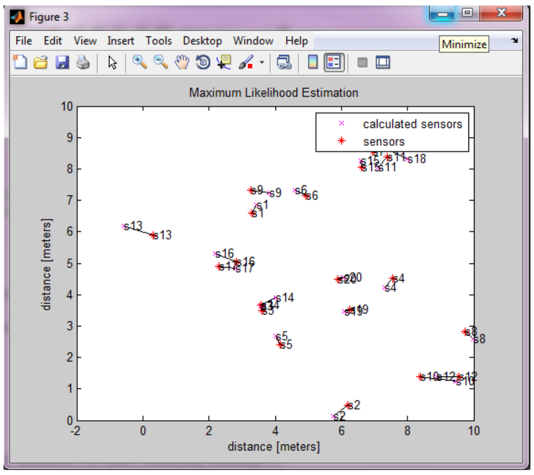


Figure 3.3b MLE, random anchors: real vs. simulated node position for 15 anchors and 20 sensors

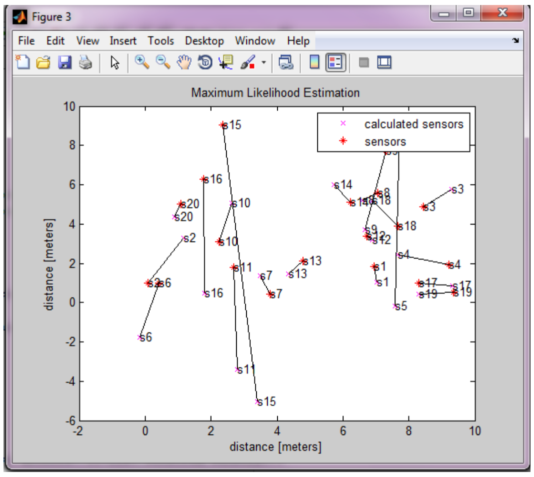


Figure 3.3c MLE results for 3 uniform anchors and 20 random sensors

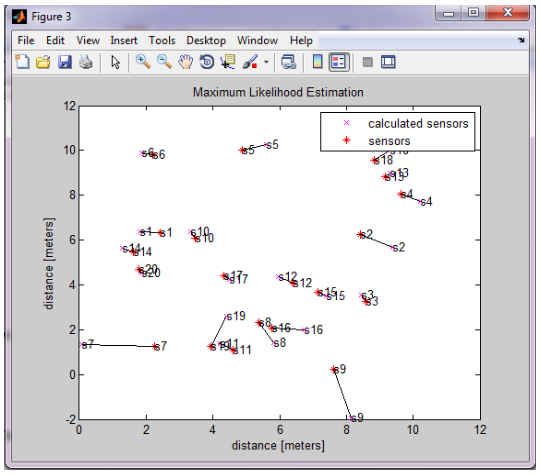


Figure 3.3d MLE results for 15 uniform anchors and 20 random sensors

For all the three anchor scenarios the average error does not have large variations when using 7 to 15 anchors. It can be said that for this interval the average error is rather stable.

However, the chart reveals that the largest average errors are obtained for the random anchor placement, while the smallest average errors are obtained for the hybrid placement. The uniform placement is in between the two configurations, somewhat closer to the hybrid placement.Up to this point the effect of different anchor configurations was studied. It is interesting to observe how the average error varies when using one configuration for the random and uniform anchors, while varying the sensor positions. For these simulations 50 different sets of sensor coordinates were used, each set being simulated for 25 times.

In Fig. 3.5 the results for random anchors are presented, while in Fig. 6 the results for uniform anchors are presented.

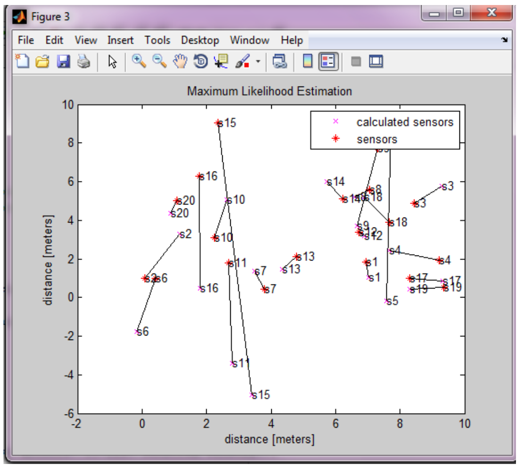


Figure 3.3e MLE results for 3 random anchors and 20 random sensors

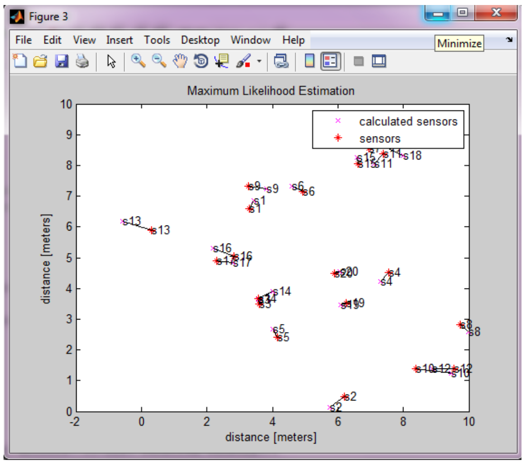


Figure 3.3f MLE results for 15 random anchors and 20 random sensors

A careful analysis of the graphs confirms the conclusions presented above:

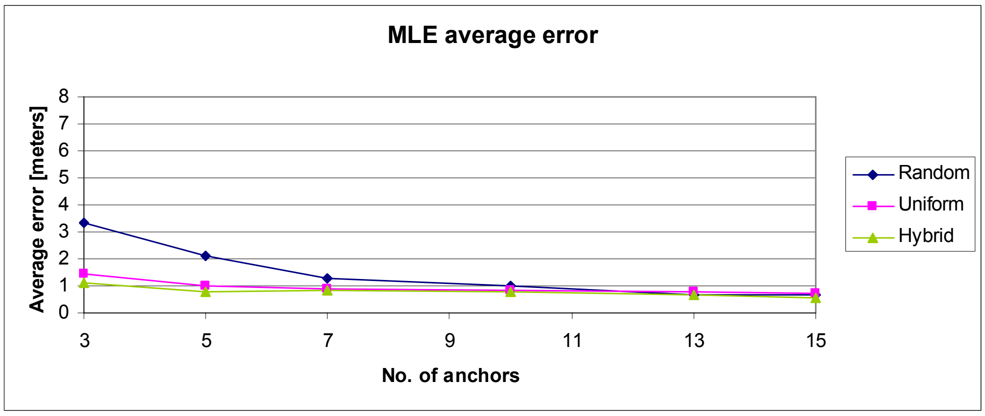


Figure 3.4a MLE average error for the considered anchor configurations

the average error for random anchors is larger than the average error for uniform anchors. In Fig. 3.5 the average error varies between 0.65 meters and 1.1 meters, while the average error in Fig. 3.6 varies between 0.6 meters and 0.87 meters. It is interesting to note that both in Fig. 3.5 and Fig. 3.6 the curves depicting the average errors have similar shapes and means of variation, in-between different boundaries.

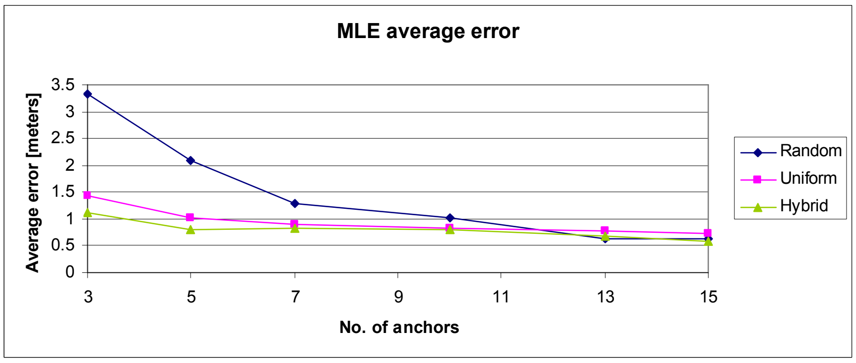


Figure 3.4b MLE average error for the considered anchor configurations: close-up

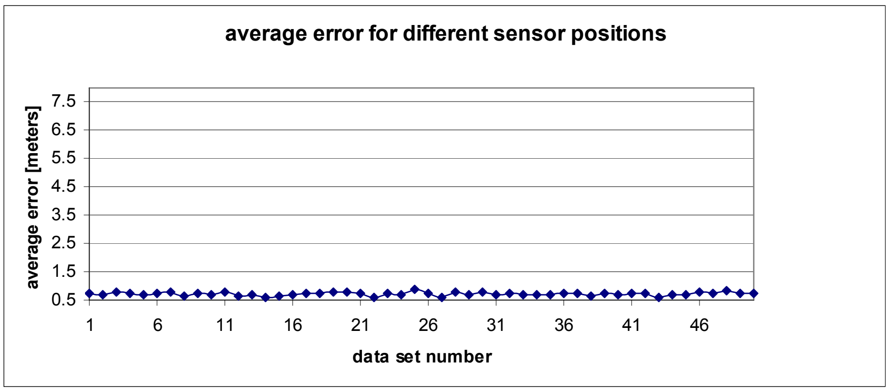


Figure 3.5: MLE average error for the considered sensor configurations and uniform anchors

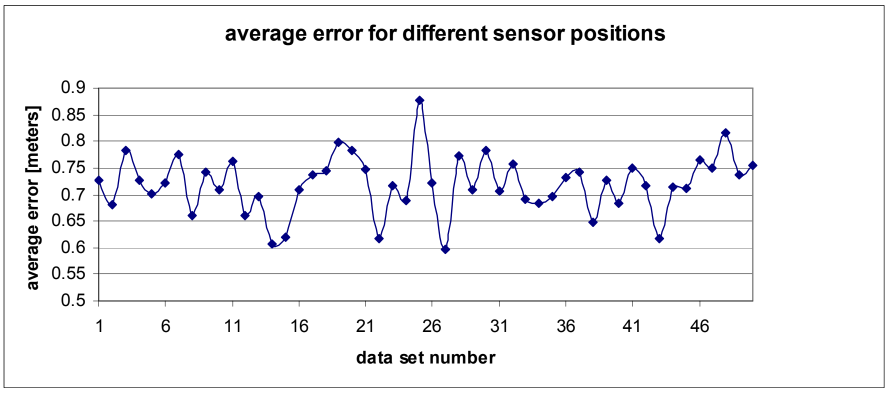


Figure 3.6: MLE average error for the considered sensor configurations and uniform anchors: close-up

3.3. Weighted Centroid Localization (WCL)

The simulations for the WCL were done in a similar manner to the simulations for the MLE algorithm. Here the same sensor deployment strategies as for the previous algorithm were used.

Table 3.2 contains the simulation results for random and uniform anchor placements.

As it can be observed from Table 3.2, the largest average errors are obtained when having randomly distributed anchors for each simulation set-up. For example, in scenario S1 the average error obtained for 3 anchors is 4.08 meters, whereas in scenario S2 the average error for the same number of anchors is 3.70 meters. All in all, for the five simulated scenarios the average error for 3 anchors varies between 3.56 meters and 4.08 meters.

Similarly, for the maximum number of uniform anchors, 15, the average error varies between 0.73 meters and 2.32 meters. For example, in scenario S1 the average error for 15 anchors is 2.32 meters, while for scenario S2 the average error is 2.02 meters. In as far as the uniform anchor placement is concerned, the average error is considerably smaller than for the random anchors. The average error for uniform anchors is almost half than the average error for random anchors.

For example, in scenario S3 an average error of 2.01 meters was obtained for 3 uniform anchors, while for 15 uniform anchors the average error obtained was of 0.87 meters. Similarly, for S5 the average error for 15 uniform anchors was 2.38 meters, while for 3 uniform anchors the average error was 0.85. The average error for the simulated scenarios varies between 1.88 meters and 2.38 meters for 15 uniform anchors and between 0.86 meters and 1.15 meters for 3 uniform anchors. Another observation that can be made by analyzing Table 3.2 is that the average error decreases as the number of anchors increases (as expected).

Table 3.2 WCL simulation results for random and uniform anchors

	Random anchors					**Uniform anchors**				
	Simulation set-up					**Simulation set-up**				
No. of anchors	**S1**	**S2**	**S3**	**S4**	**S5**	**S1**	**S2**	**S3**	**S4**	**S5**
	Average error [meters]					**Average error [meters]**				
3	4.08	3.70	3.69	3.56	3.58	2.32	2.02	2.01	1.88	2.38
5	3.26	2.72	2.72	2.93	2.75	1.96	1.52	1.52	1.58	2.14
7	3.00	2.35	2.35	2.69	2.41	1.31	1.23	1.25	1.23	1.19
8	3.09	2.34	2.34	2.54	2.35	1.37	1.30	1.30	1.25	1.34
9	3.06	2.35	2.34	2.41	2.37	1.16	1.10	1.10	1.11	1.16
10	2.38	1.68	1.68	1.50	1.69	1.08	0.94	0.94	0.93	1.05
11	2.47	1.63	1.63	1.48	1.73	1.04	0.91	0.90	0.86	0.99
12	1.47	1.13	1.13	1.06	1.26	1.04	0.89	0.90	0.84	0.93
13	1.48	1.18	1.18	1.04	1.26	1.02	0.89	0.89	0.81	0.92
14	1.41	1.10	1.10	0.96	1.23	1.05	0.88	0.86	0.75	0.87
15	1.37	1.12	1.21	0.98	1.15	0.96	0.86	0.87	0.73	0.87

Based on the above mentioned observations the necessity to reduce the average error for randomly distributed anchors arises. In order to achieve this, a new anchor placement strategy was developed. The new anchor placement was called a *hybrid anchor placement*.

The results obtained after simulating each of the five scenarios 25 times each are listed in Table 3.3. When looking at Table 2 and Table 3 some observations can be made. Firstly, when comparing the average error for randomly distributed anchors and for hybrid distributed anchors a considerable improvement is observed. The error reduction is more significant when using a small number of anchors then when using a large number of anchors. For example, when examining scenario S4 the average error for 3 random anchors is 3.56 meters, whereas for 3 hybrid anchors the average error is 2.20 meters. The same situation occurs for 7 anchors: when random the average error is 2.54 meters, when hybrid the average error is 1.48 meters. In as far as the maximum number of possible anchors, 15, the difference between the obtained average errors is smaller. For the considered scenario, when having random anchors the average error is 0.98 meters and when having hybrid anchors the average error is 1.06 meters.When comparing the hybrid placement with the uniform placement the situation is quite similar. For the number of anchors varying between 3 and 7 there is a more significant error reduction than for the number of anchors varying between 7 and 15. From 7 anchors onwards the difference between the obtained average errors is quite small.

It can also be observed that the largest errors are obtained for the random anchor distribution. When comparing the uniform distribution with the hybrid distribution it can be seen that the errors obtained for the hybrid distribution are larger, but the difference is not that significant.

All in all it can be concluded by stating that the hybrid anchors placement reduces considerably the errors obtained by randomly placing anchors. This is due to the 4 anchors placed in the corners of the room that always assure that those areas are covered and then to the remaining 11 random anchors that assure coverage in different sections of the room.

Table 3.3 WCL simulation results for hybrid anchors

No. anchors	**Simulation set-up average error [meters]**				
	S1	**S2**	**S3**	**S4**	**S5**
3	2.58	2.66	2.19	2.20	2.56
4	1.29	1.28	1.35	1.34	1.49
5	1.38	1.53	1.42	1.41	1.67
6	1.40	1.64	1.51	1.51	1.83
7	1.50	1.43	1.49	1.48	1.54
8	1.44	1.40	1.33	1.32	1.34
9	1.33	1.25	1.30	1.30	1.20
10	1.38	1.19	1.18	1.17	1.15
11	1.13	1.05	1.11	1.12	1.04
12	1.15	1.10	1.11	1.11	1.00
13	1.17	1.09	1.08	1.07	1.00
14	1.12	1.06	1.06	1.07	1.02
15	1.13	1.11	1.06	1.06	1.02

However, when comparing the hybrid approach to the uniform approach, it can be observed that the average errors obtained for the hybrid approach are larger than the average errors obtained for the uniform approach. But, the difference is not that big, and most importantly, the purpose of reducing the average error for randomly distributed anchors was achieved. In Fig. 3.7, 3.8, 3.9, 3.10 and 3.11 the errors given in Tables 3.2 and 3.3 are illustrated by means of graphs. When comparing the hybrid placement with the random placement the greatest improvements can be observed. These improvements are obvious when the number of anchors varies between 3 and 11. From 11 to 15 anchors the average errors are quite close to each other.

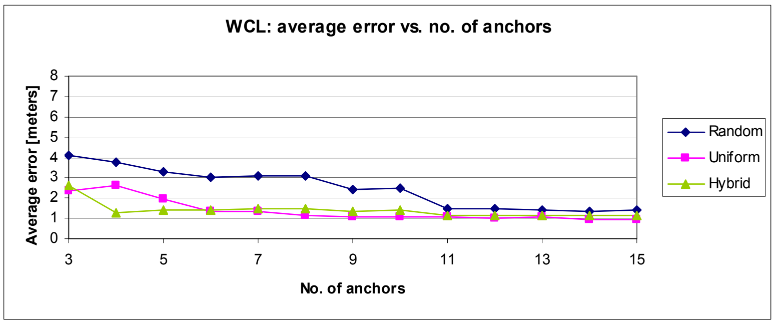


Figure 3.7a: WCL average error for scenario S1

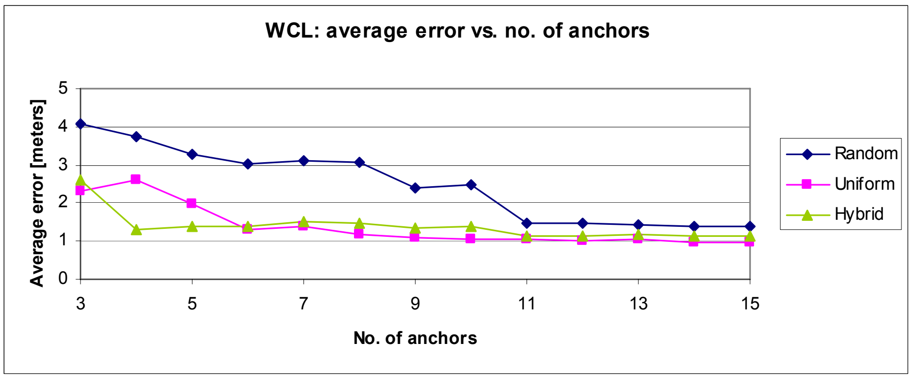


Figure 3.7b: WCL average error for scenario S1: close-up

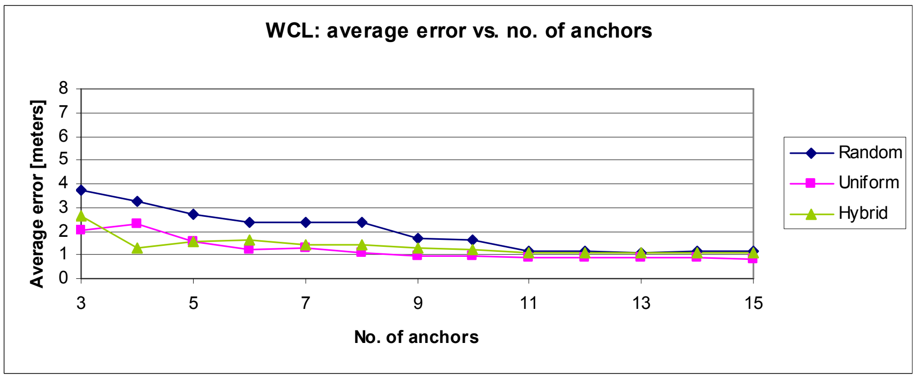


Figure 3.8a: WCL average error for scenario S2

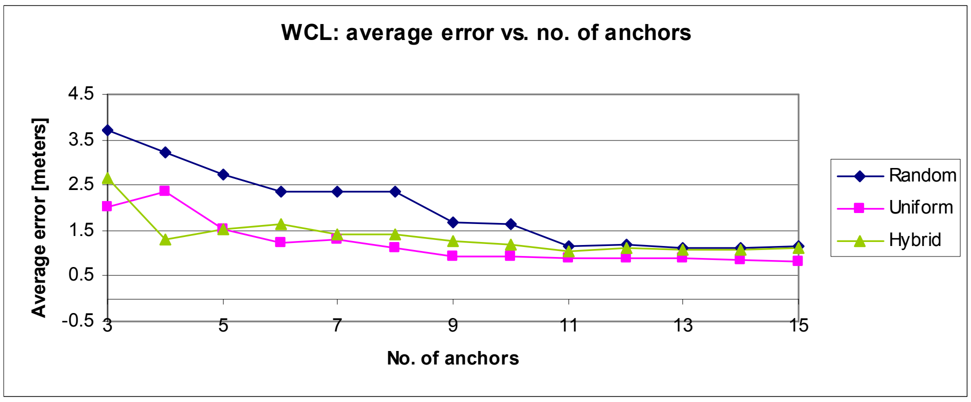


Figure 3.8b: WCL average error for scenario S2: close-up

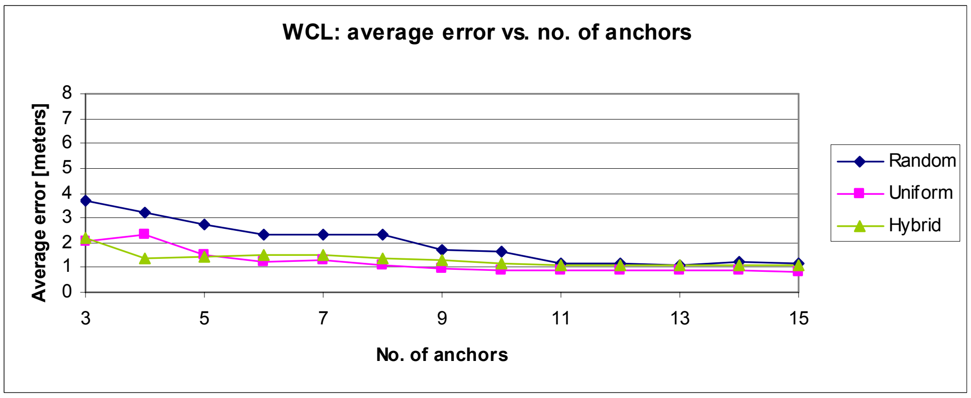


Figure 3.9a: WCL average error for scenario S3

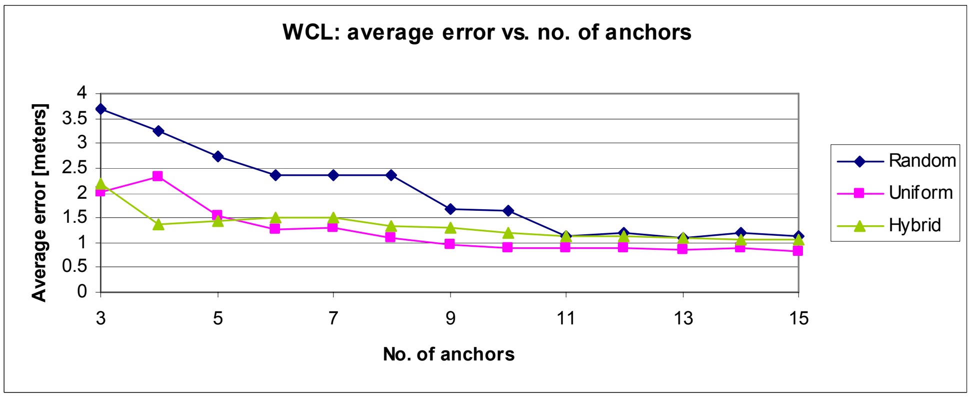


Figure 3.9b: WCL average error for scenario S3: close-up

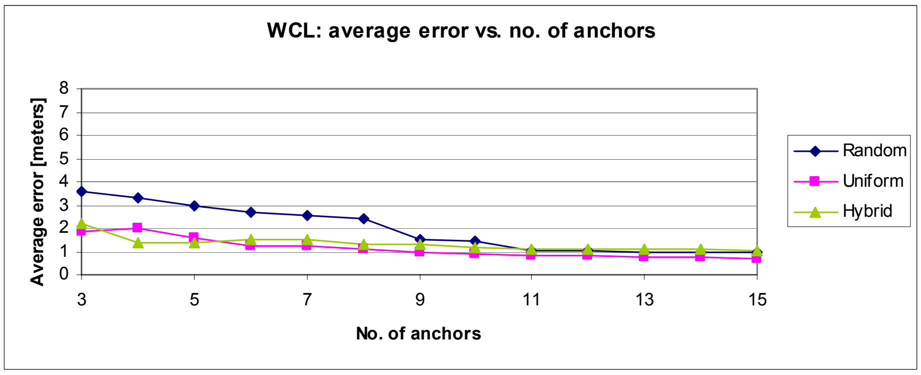


Figure 3.10a: WCL average error for scenario S4

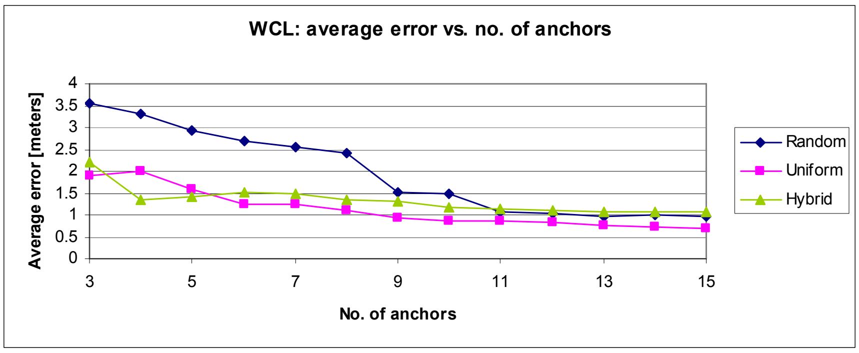


Figure 3.10b: WCL average error for scenario S4: close-up

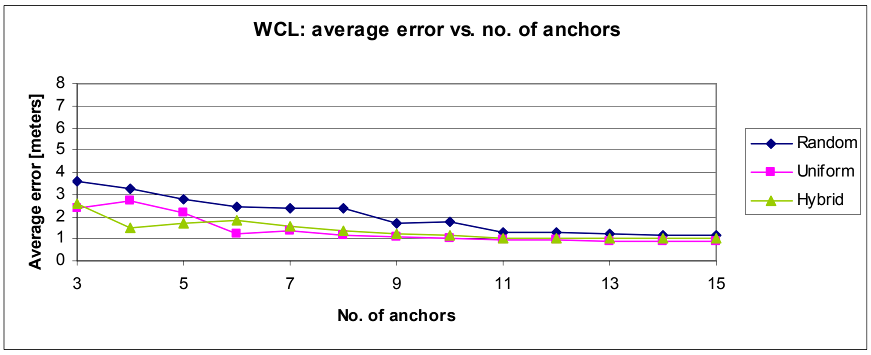


Figure 3.11a: WCL average error for scenario S5

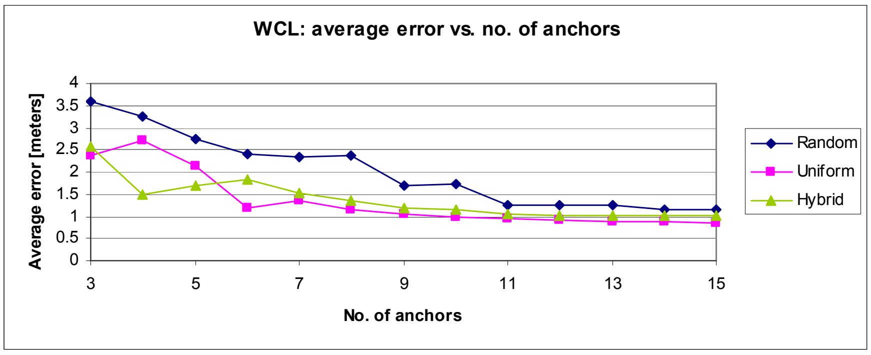


Figure 3.11b: WCL average error for scenario S5: close-up

3.4. Iterative Trilateration

The first simulated scenarios were for the random and uniform anchor placements. The obtained results are listed in Table 3.4.

When analyzing Table 3.4 a few observations can be made. First of all, the average error obtained for 3 random anchors is considerably larger than the average error obtained for 3 uniform anchors. Taking scenario S1 as an example, the average error obtained for 3 random anchors is 6.49 meters, while the average error obtained for 3 uniform anchors is 0.81 meters. Things are very similar for the other 4 scenarios. When considering 15 anchors, the obtained average errors for the random and uniform anchor placements are very similar. Looking at S1, for random anchors the obtained error is of 0.26 meters, while for uniform anchors the average error is 0.25 meters.

Even though the uniform anchor placement gives considerably better results for iterative trilateration than the random placement, the hybrid anchor deployment will be simulated and the results will be compared. The results obtained using hybrid deployments are listed in Table 3.5.

A conclusion can be drawn: firstly when comparing the average errors for hybrid and random anchors the improvements are considerable. However, when comparing the average errors for hybrid and uniformly distributed anchors the results are alike. That is, when looking at scenario S3, the average error for 3 uniform anchors is 0.78 meters, while for 3 hybrid anchors the error is 1.03 meters. For 15 uniform anchors the average error is 0.27 meters, while for 15 hybrid anchors the average error is 0.27 meters. These above stated conclusions can be observed better by looking at Fig. 3.12-3.15. These figures illustrate the obtained errors for all 3 anchor configurations. From Fig. 3.12-3.15 it can be observed that the average error obtained for random anchors is larger than the average error obtained for uniform or hybrid placement. But the error obtained for uniform and hybrid placement is very similar, as the lines denoting the average error for both anchor placements overlap for all the scenarios. Having observed this, it can be concluded that for the iterative trilateration algorithm, in order to obtain good results, either the hybrid

Table 3.4 Iterative trilateration error for random and uniform anchors

No. anchors	Random anchors average error [meters]					Uniform anchors average error [meters]				
	Simulation set-up					Simulation set-up				
	S1	**S2**	**S3**	**S4**	**S5**	**S1**	**S2**	**S3**	**S4**	**S5**
3	6.49	5.62	5.10	5.38	5.78	0.81	0.69	0.78	0.74	0.89
4	5.96	5.02	6.54	4.88	5.29	0.82	0.65	0.69	0.78	0.80
5	4.90	4.07	3.91	4.21	4.52	0.76	0.51	0.53	0.60	1.41
6	4.06	2.88	3.11	3.17	3.68	0.48	0.45	0.50	0.49	0.46
7	3.82	2.34	2.59	2.34	3.37	0.41	0.40	0.45	0.41	0.40
8	3.76	2.56	2.74	2.44	3.23	0.39	0.37	0.39	0.37	0.37
9	2.25	1.29	1.95	1.21	2.28	0.34	0.33	0.36	0.36	0.35
10	1.43	1.11	1.54	0.91	1.78	0.34	0.32	0.36	0.33	0.33
11	0.32	0.28	0.31	0.30	0.30	0.31	0.31	0.33	0.32	0.30
12	0.33	0.28	0.38	0.37	0.30	0.30	0.29	0.31	0.30	0.30
13	0.27	0.28	0.27	0.28	0.26	0.28	0.28	0.30	0.28	0.29
14	0.28	0.25	0.28	0.26	0.27	0.28	0.27	0.28	0.27	0.28
15	0.26	0.25	0.27	0.25	0.24	0.25	0.25	0.27	0.25	0.26

anchor placement or the uniform anchor placement can be used. However, in real life it is more likely to encounter hybrid anchor distributions than uniform.

Table 3.5 Iterative trilateration error results for hybrid anchors

No. anchors	Hybrid distributed anchors average error [meters]				
	Simulation set-up				
	S1	**S2**	**S3**	**S4**	**S5**
3	1.10	1.02	1.03	0.77	0.89
4	0.63	0.64	0.65	0.67	0.66
5	0.56	0.52	0.58	0.56	0.57
6	0.52	0.47	0.52	0.48	0.49
7	0.46	0.47	0.46	0.44	0.45
8	0.42	0.44	0.41	0.39	0.39
9	0.37	0.39	0.37	0.37	0.36
10	0.39	0.33	0.47	0.33	0.35
11	0.32	0.31	0.34	0.34	0.32
12	0.32	0.29	0.33	0.29	0.31
13	0.30	0.28	0.31	0.29	0.29
14	0.28	0.27	0.30	0.26	0.29
15	0.27	0.25	0.27	0.27	0.28

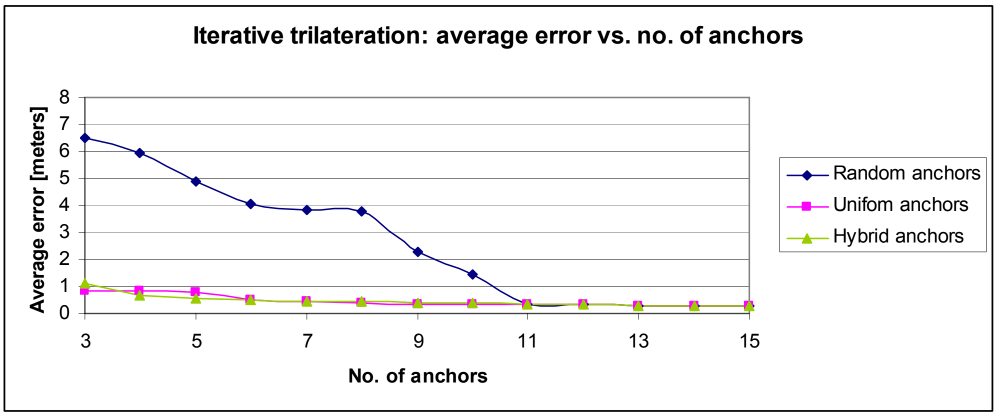


Figure 3.12: Iterative trilateration average error for scenario S1

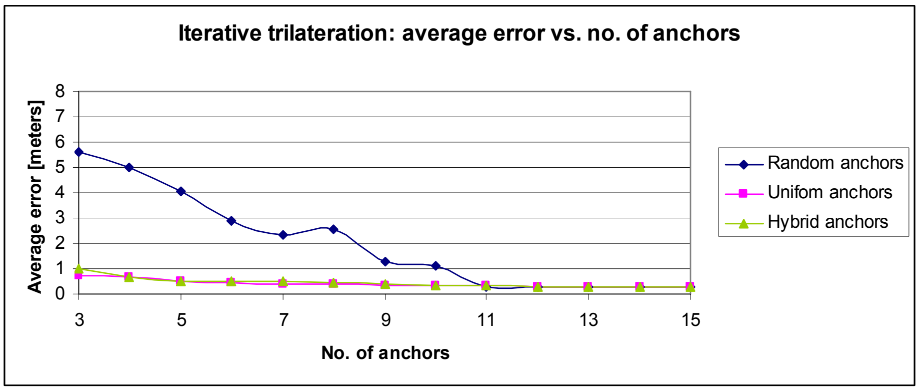


Figure 3.13: Iterative trilateration average error for scenario S2

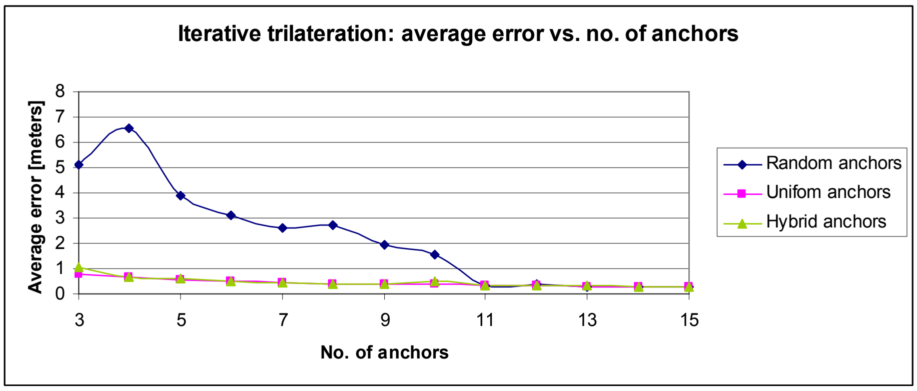


Figure 3.14: Iterative trilateration average error for scenario S3

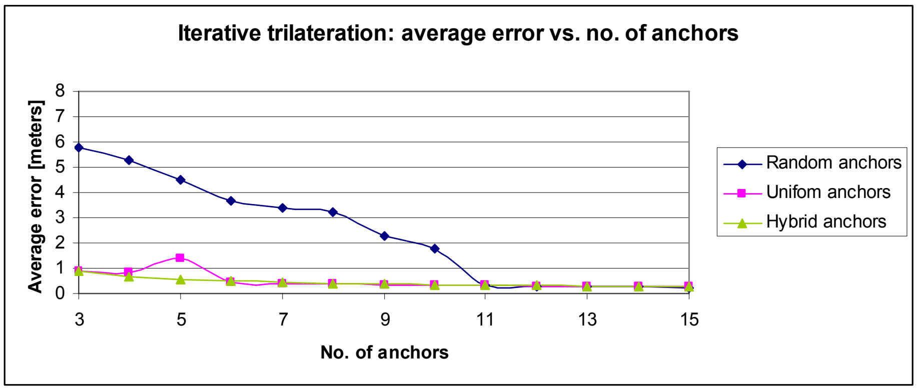


Figure 3.15: Iterative trilateration average error for scenario S5

3.5. Malguki Spring

The theoretical foundation of the algorithm revealed that the results are influenced by some parameters such as the initial value of the sensor position, the conversion parameter, as well as the overall error. For simulation purposes the author varied the parameters as follows, as stated in [Ari04]

γ	$e_{threshold}$
0.1	5
0.01	5
0.1	0.5
0.01	0.5

In Fig. 3.16 and Fig. 3.17 the average error vs. sensor number is depicted when using random anchors. After analyzing the two graphs a first observation can be made: the obtained average error for set 1 is larger than the obtained average error for set 5. This is due to the influence of the initial value of the sensor position over the algorithm results: the better results are obtained for an initial value closer to the true sensor position. The influence of the algorithm parameters is straightforward when analyzing the two graphs.

In Fig. 3.18 the results obtained for a different variation of the algorithm results is presented. In comparison with the previous situations, there are no large variations of the algorithm results.

A comparison can be made between the the author’s results and the results obtained by other authors, such as Desai and Turelli [5] or Arias et. al [4]. The results obtained here are better than those of Desai and Turelli and similar to those of Arias et. al.

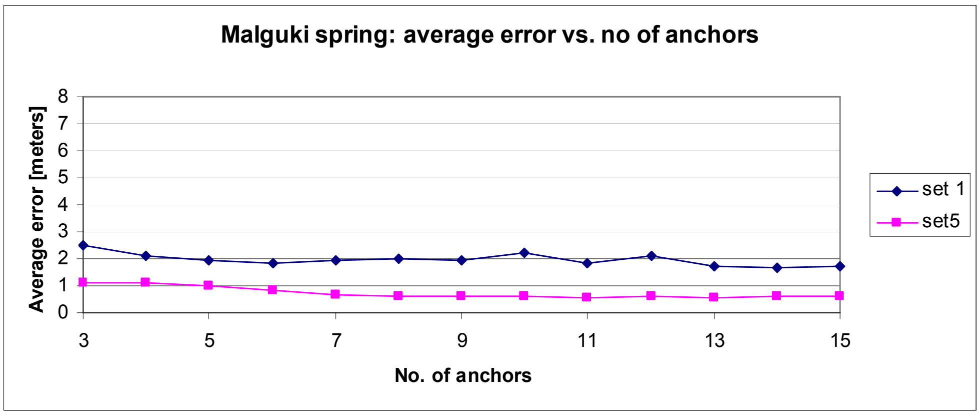


Figure 3.16a: Random anchors: average error vs. sensor number for gama=0.1 and threshold=5

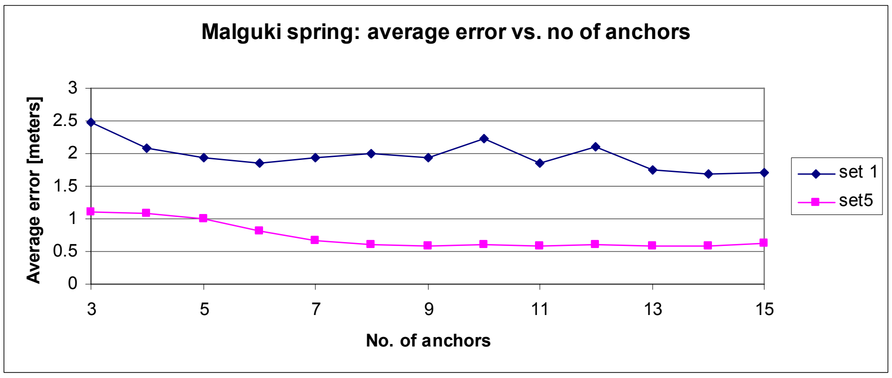


Figure 3.16b: Random anchors: average error vs. sensor number for gama=0.1 and threshold=5 : close-up

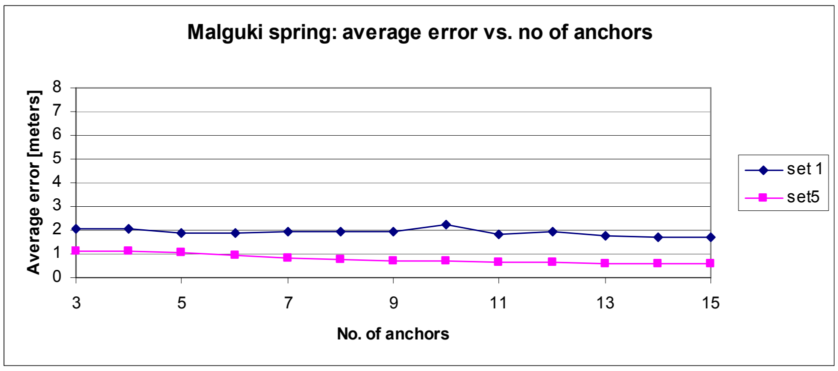


Figure 3.17: Random anchors: average error vs. sensor number for gama=0.01 and threshold=5

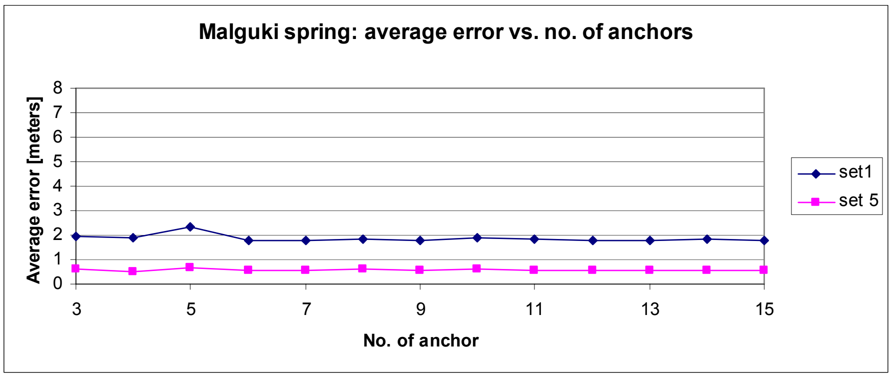


Figure 3.18: Random anchors: average error vs. sensor number for gama=0.1 and threshold=0.5

For uniform anchors the first results are depicted in Fig. 3.19. In Fig. 3.19 the average error is higher than in Fig.3.16, even though the simulation conditions are identical. This is again due to the initial value of the sensor position. In Fig. 16 the average error takes values of 0.6-1.1 meters, whereas in Fig. 3.5 the average error has values of 0.5-1 meter.

For set 5 the average error depicted in Fig.3.20 appears to stabilize around 0.5 meters, no matter the number of used anchors. The error is more stable when compared to graph in Fig. 3.18. Even more, when comparing the same graphs as before, for 3 to 7 anchors, the average error in Fig. 18 is almost double than the average error in Fig. 3.20.

In as far as the results presented in Fig. 3.18, no significant changes are observed.

For the **hybrid deployment** the first results for this approach are presented in Fig. 3.19. An immediate improvement can be observed in as far as set 1 is concerned. When using 3 anchors the average error was reduced from 2.5 meters to 1.8 meters, being quite stable around this value. A certain stability of the average error can be observed for set 5: the average error varies between 0.6 and 1.1 meters, being more stable when using more than 7 anchors.

The graph in Fig. 3.20 depicts lower average errors for hybrid deployments than for the other two anchor configurations. For set 1 an average error of 1.5 meters is obtained when using 3 anchors, while for set 5 the average errors is reduced to 0.4 meters when using 3 to 5 anchors (in previous situations the average error was 1 meter).

In Fig. 3.21 no remarkable changes are noticed when comparing the results to the previous ones.

In conclusion09, the hybrid anchor placement can be regarded as an improvement for the uniform and random anchor deployment. The hybrid deployment proves to be more advantageous when using smaller numbers of anchors.

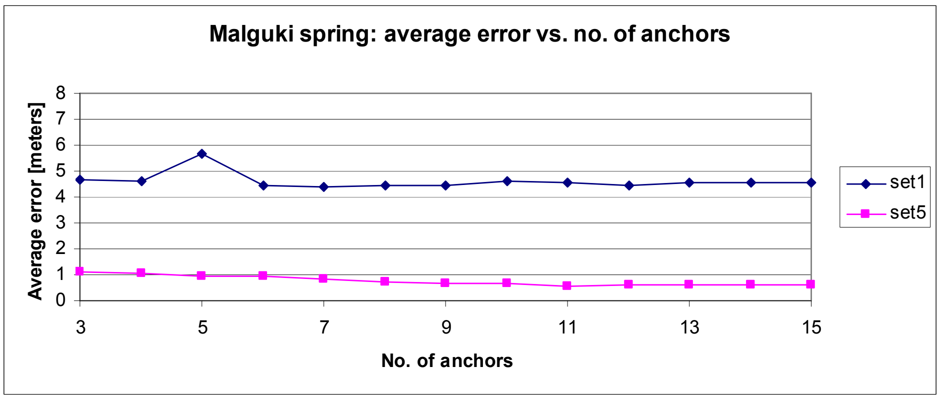


Figure 3.19a: Uniform anchors: average error vs. sensor number for gama=0.1 and threshold=5

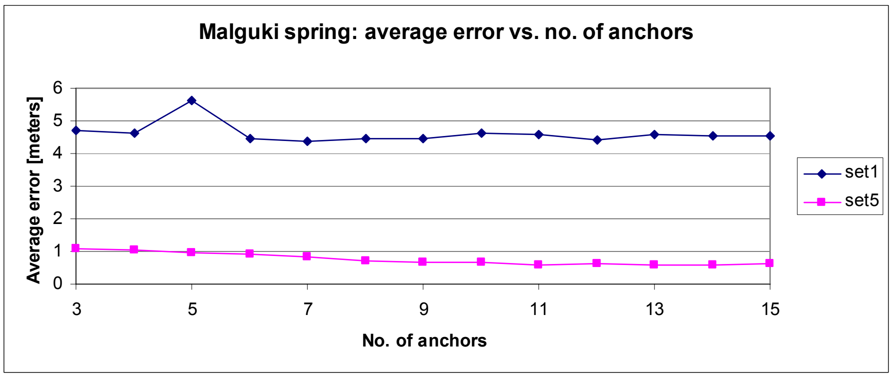


Figure 3.19b: Uniform anchors: average error vs. sensor number for gama=0.1 and threshold=5: close-up

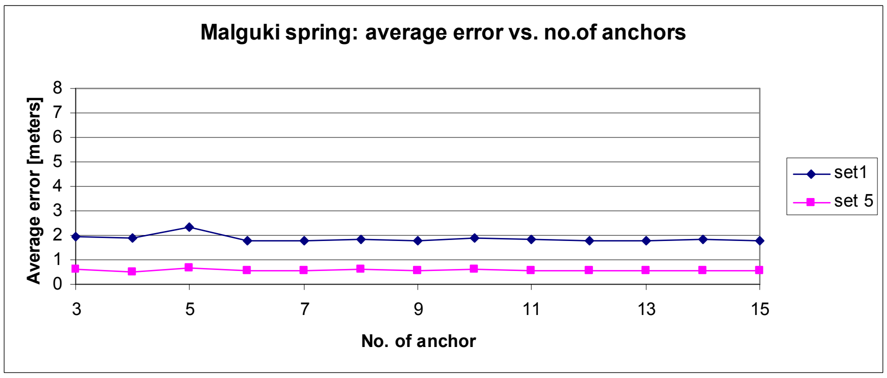


Figure 3.20a: Uniform anchors: average error vs. sensor number for gama=0.1 and threshold=0.5

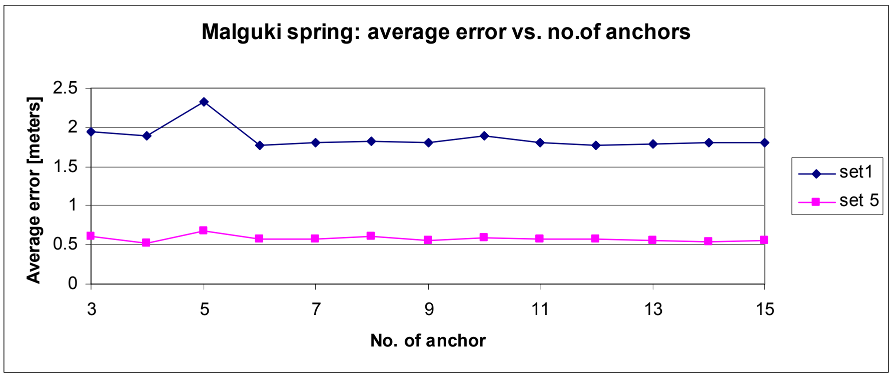


Figure 3.20b: Uniform anchors: average error vs. sensor number for gama=0.1 and threshold=5: close-up

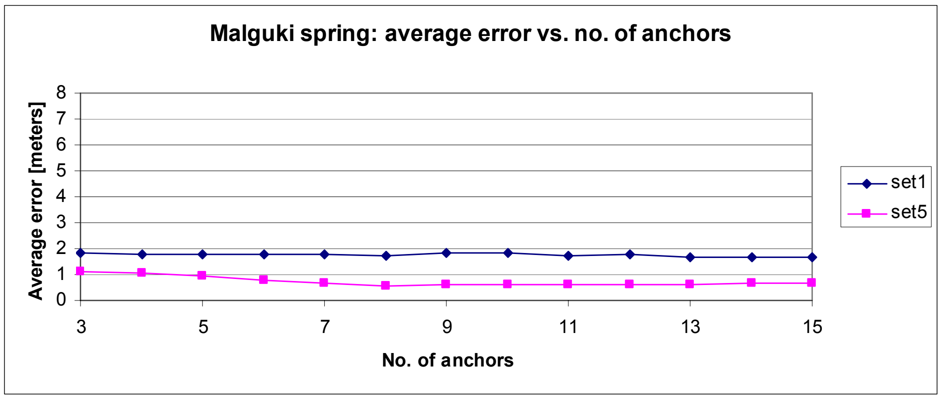


Figure 3.21a: Hybrid anchors: average error vs. sensor number for gama=0.1 and threshold=5

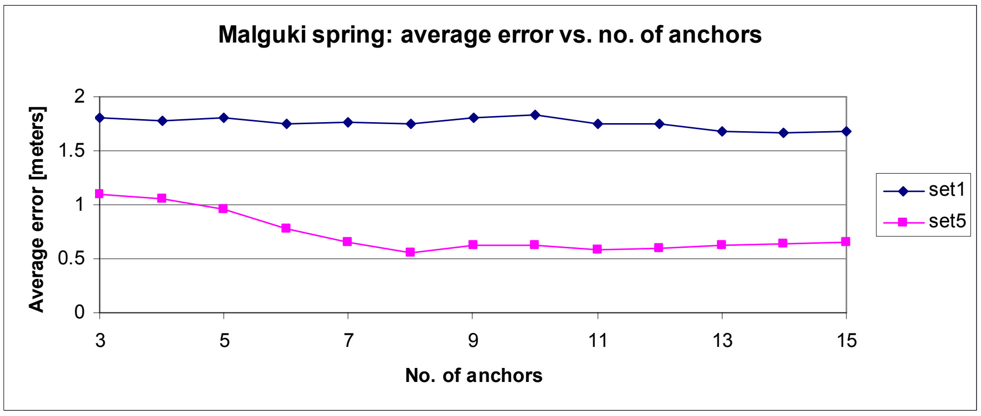


Figure 3.21b: Hybrid anchors: average error vs. sensor number for gama=0.1 and threshold=5

As pointed out throughout presentation of the results, the algorithm is significantly influenced by the value of the initial sensor position. This value is very important due to the fact that the Malguki spring can be regarded as an optimization algorithm, and, it is a well known issue that all optimization algorithms are largely influenced by the initial guess that is used to start them. In the following lines a proof of the influence of the initial guess over the algorithm results is made.

Table 3.6 presents the obtained average errors for 3 different sensor configurations (other than sets 1 and 5) when using random anchors. The simulations parameters are $\gamma = 0.01$ and $e_{threshold} = 0.5$.

By analyzing Table 3.6 it can be observed that the right choice of the starting value yields smaller average errors than the inappropriate choice. This is valid for any of the considered scenarios. Therefore, in order to obtain a minimum error the initial value $\vec{s}_0$ must be carefully chosen. Even though the error threshold and the value for γ is altered, the influence of the starting values is significant over the obtained results: if the starting value is correctly chosen the obtained errors are considerably smaller than those obtained for an incorrect choice of $\vec{s}_0$.

Table 3.6 Influence of the algorithm starting value over the algorithm results

No. anchors	**Inappropriate choice of starting value- average error [meters]**			**Appropriate choice of starting value- average error [meters]**		
	Set 6	**Set 8**	**Set 9**	**Set 6**	**Set 8**	**Set 9**
3	4.07	4.53	4.66	0.99	0.93	0.44
10	4.40	4.99	4.27	0.95	0.94	0.50
15	4.21	4.70	4.04	0.95	0.62	0.49

The Malguki spring algorithm was simulated using three different set-ups for the anchor nodes. An average error was calculated and the results were compared. The results obtained for uniform anchors were better than those obtained for random anchors. However, there is a necessity to reduce the error when using ab small numbers of anchors. The reduction was achieved using hybrid anchors. The error decreased when using 3 to 6 anchors and it became more stable form 7 anchors onwards. The hybrid placement can be regarded as an error reduction method. The obtained average errors are comparable to the errors encountered in literature. The influence of the starting algorithm values over the obtained results was discussed

3.6. Modified Multidimensional Scaling (MMDS)

The results obtained using MMDS are presented in this section. For the simulations the same scenarios as before are considered and the results are presented in a similar manned.

In Table 3.7 MMDS results for all of the considered scenarios are presented.

Table 3.7 Wireless Sensor Networks Node Models Error

No. of anchors	Average error [meters]		
	Random Anchors	Uniform Anchors	Hybrid Anchors
3	4.01	3.77	3.80
5	3.54	3.72	3.60
7	4.02	3.51	3.73
10	3.98	3.48	3.74
15	3.60	3.49	3.51

The errors obtained are very similar for all of the three scenarios. The values range between 3 and 4 meters, such values being large. In these conditions the algorithm is not very reliable. The graphical depiction of the table results can be found in Figs. 3.22-3.24. Error variations in the graphs are high, the curves bind. However for uniform and hybrid anchors the graphs do not present as many peaks as for the random anchor placement. This proves that, even though the errors are still high, a slight improvement has been made. The error is maximum for node number 4 while for the rest of the considered nodes the errors are rather uniform.

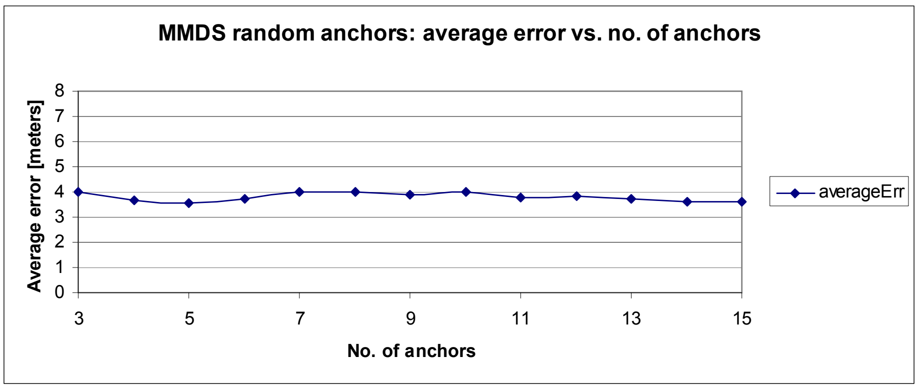


Figure 3.22a: MMDS results for random anchors

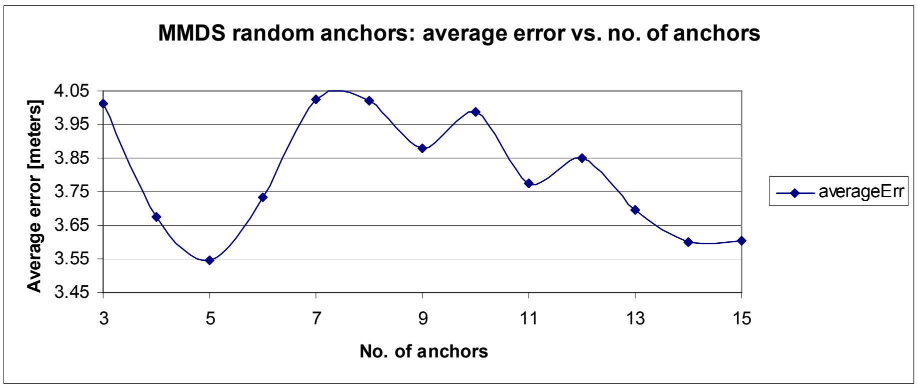


Figure 3.22b: MMDS results for random anchors: close-up

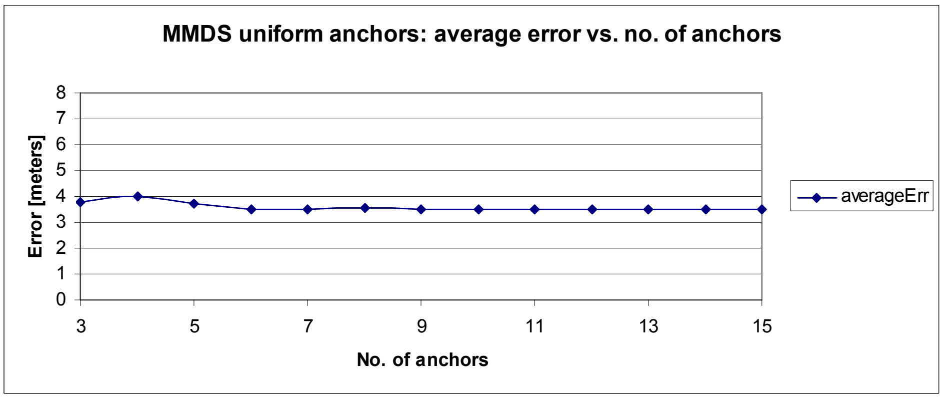


Figure 3.23a: MMDS results for uniform anchors

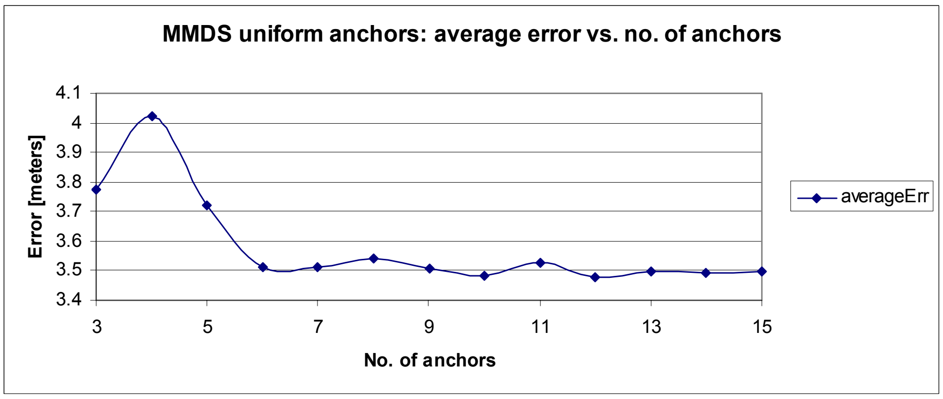


Figure 3.23b: MMDS results for uniform anchors: close-up

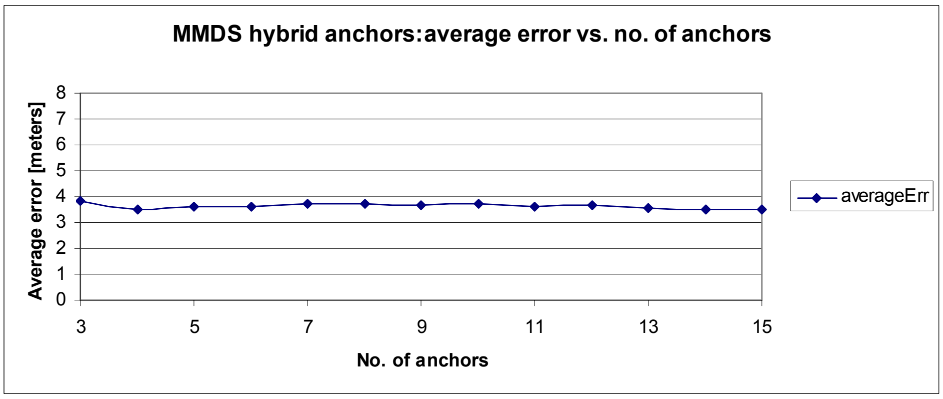


Figure 3.24a: MMDS results for hybrid anchors

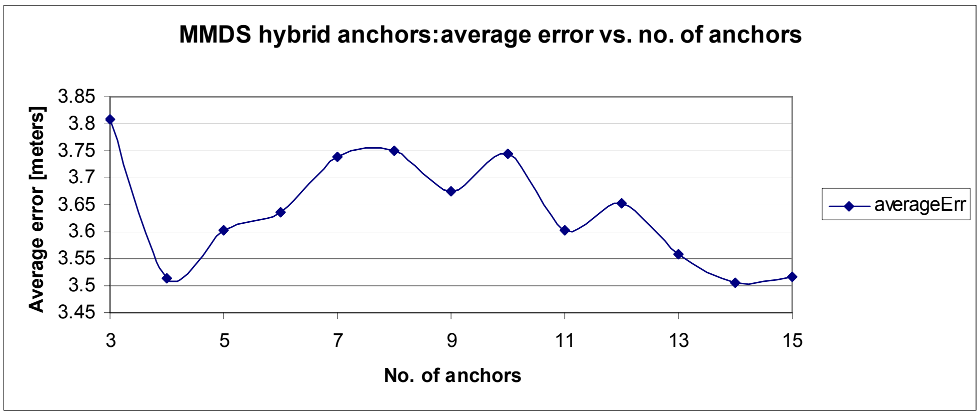


Figure 3.24b: MMDS results for hybrid anchors: close-up

3.7. Conclusions

In this chapter the author presented original simulation results. The code for all the algorithms surveyed in Chapter 2 was written by the author in Matlab. The simulation scenarios studied by the author were the random and the uniform configurations. Since the results (positioning error) provided by the algorithm required some improvements the author came up with a new simulation scenario. This scenario was named hybrid and to the author's knowledge such a configuration was not encountered anywhere in literature. The idea for this novel scenario came at a careful analysis of the error obtained using random and uniform anchors. The hybrid configuration combined the two classical scenarios in the sense that four anchors are placed at an equal distance from the corners of the space, while the rest are deployed randomly in the remaining surface. Such an arrangement served the author's purpose, bringing, for most of the studied algorithms the desired error reduction.

A comparative analysis of the algorithms studied in this chapter can be found in Table 3.8.

For the MLE the results obtained for random anchors are the highest: for 15 anchors an error of 3.33 meters is obtained, while for 3 anchors the error is 0.64 meters. The random anchors provide better results than the uniform anchors: the error becomes 1.44 meters for 15 anchors (almost half than before) and 0.72 meters for 3 anchors (similar to the random anchors). However the aim is to reduce the error of the random anchors without losing the randomness. This is achieved through the hybrid placement. The results obtained are 1.13 meters for 15 anchors and 0.58 meters for 3 anchors. The results are better than for both random and uniform placement and thus the goal was achieved.

Last but not least, the worst results are obtained when using the MMDS. For this algorithm the errors are very high when using random anchors and, even though they reduce as the number of anchors increases, the difference is not significant. Moreover, the other two random placement strategies do not improve the obtained results. Judging by this, it is not recommended to use this algorithm for node position determination.

Table 3.8 Wireless Sensor Networks Error Comparison

Algorithm	No. anchors	Average error [meters]		
		Random anchors	Uniform anchors	Hybrid anchors
MLE	3	3.33	1.44	1.13
	10	1.02	0.83	0.80
	15	0.64	0.72	0.58
WCL	3	4.08	2.32	2.58
	10	2.47	1.04	1.38
	15	1.38	0.95	1.13
Iterative trilateration	3	6.49	0.81	1.10
	10	1.43	0.34	0.39
	15	0.26	0.25	0.27
MMDS	3	4.01	3.77	3.80
	10	3.98	3.48	3.74
	15	3.60	3.49	3.51
Malguki spring	3	2.47	2.16	1.80
	10	2.23	2.41	1.83
	15	1.69	2.45	1.68

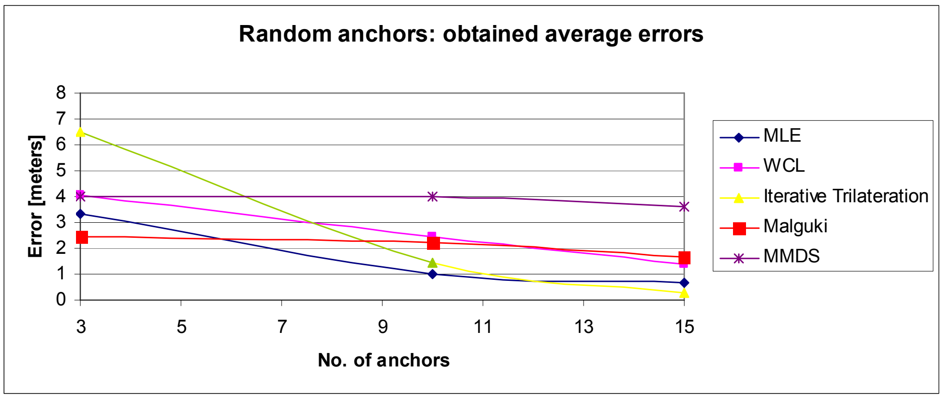


Figure 3.25: The obtained average errors for random anchors

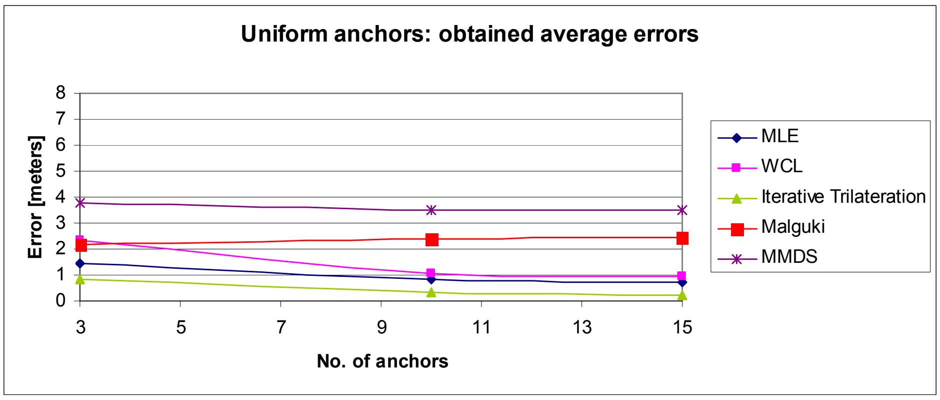


Figure 3.26a: The obtained average errors for uniform anchors

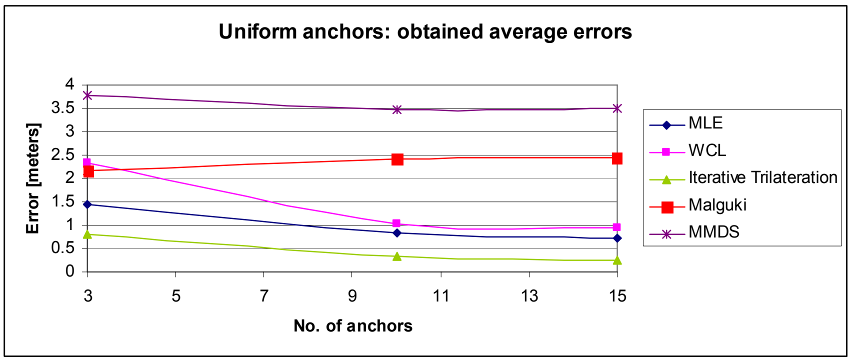


Figure 3.26b: The obtained average errors for uniform anchors: close-up

Considering the above and Table 3.8 a few observations can be made: the obtained
errors decrease as the number of anchors increase. Then, the largest
errors are generally obtained for random anchors placements, whereas improvements are obtained for uniform and hybrid placements.

The errors for the WCL algorithm are 4.08 meters for 15 random anchors, and 1.38 meters for 3 random anchors. The uniform placement reduces the errors to almost half for 15 anchors, while for 3 anchors the error reduction is obvious but not that significant. The results for the hybrid placement are similar to the uniform placement.

The errors provided by the iterative trilateration are very high for the uniform anchors, especially when using 15 anchors an error of 6.49 meters is obtained. When having 3 random anchors the error is 0.26 meters. The uniform and hybrid placements provide similar errors that are considerably reduced for the maximum number of used anchors. In Table 3.9 a comparison of the results obtained by the author with the results obtained by other authors is made. This comparison is made for the random placement of anchors, since [Des07] use this type of placement.

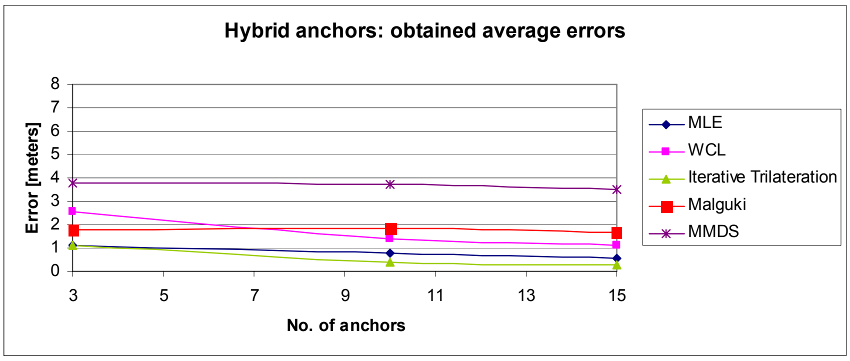


Figure 3.27a: The obtained average errors for hybrid anchors

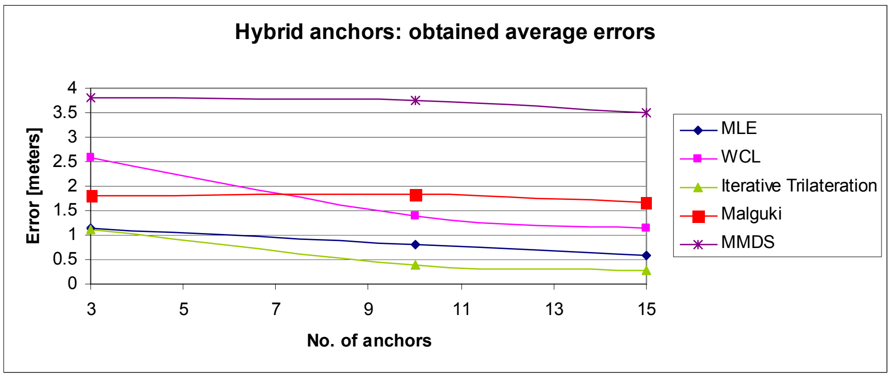


Figure 3.27b: The obtained average errors for hybrid anchors: close-up

Table 3.9 Wireless sensor networks error comparison with [Des07]

Algorithm	No. of anchors	Error RR [meters]- Random anchors	Error [Des07] [meters]
MLE	3 5	3.33 2.10	5.2 5.5
Malguki Spring	3 5	2.47 1.2	4.1 3.5

Since [Des07] use a maximum number 7 anchors Table 3.9 compares the results obtained by the author and [Des07] up to a maximum number of 5 anchors. The algorithms presented in Table 3.9 are MLE and Malguki spring. The results presented by [Des07] for both algorithms reveal larger errors for the same number of anchors. It is interesting to note that the results provided by [Des07] for random anchors are larger than the author's not only for the random placement of anchors.

Table 3.10 presents a comparative analysis of WCL results obtained by the author and [Blu07]. Since [Blu07] uses 4 anchors for the analysis, the results will be presented for this number of anchors.

Table 3.10 Wireless sensor networks error comparison with other [Blu07]

Algorithm	No. of anchors	Error RR [meters]- Uniform anchors	Error [Ari04] [meters]
WCL	4	2.1	2.5-5.3

The error obtained by [Blu07] for WCL are larger than the errors obtained by the author when using WCL. The results are presented for a uniform anchor placement.

In Table 3.11 an analysis of MMDS results is presented. This analysis is performed for [Des07] and the author's results.

The errors obtained by [Des07] for 3 anchors are larger than the author's results for the same number of anchors. For 5 anchors the errors obtained by [Des07] and the author are similar

Table 3.11 Wireless sensor networks error comparison with [Des07]

Algorithm	**No. of anchors**	**Error RR [meters]-Random anchors**	**Error [Des07] [meters]**
MMDS	3 5	4.01 3.54	4.5 3.5

Malguki spring algorithm is analyzed in Table 3.12. Since [Ari07] use a number of 4 to 14 anchors the comparison is done for 4 and 10 anchors. Table 3.12 shows that the author obtained smaller errors when using 4 anchors than [Ari07]. The situation is maintained for 10 anchors.

Table 3.12 Wireless sensor networks error comparison with [Ari07]

Algorithm	**No. of anchors**	**Error RR [meters]-Random anchors**	**Error [Ari07] [meters]**
MMDS	4 10	2.5 2.5	4.1 3

The algorithms studied in this chapter have different complexities. The simplest ones use direct calculus while the most complicated ones make use of optimization methods. This has impact on the implementation complexity and it represented a challenge. As pointed out throughout the chapter the author obtained better positioning errors than other authors have obtained in similar conditions.

3.8. Contributions

Since this chapter is focused on the implementation of the algorithms introduced in Chapter 2, the author's main contributions to this chapter are practical (applicative) contributions.

One contribution is the Matlab original simulations: the code written to implement the algorithms as well as the implementation of the pathloss model. From this code the second contribution is obtained: the original simulation scenario: the original hybrid set-up, which, to the author's knowledge was not encountered anywhere in literature

Another contribution is the comparative anlysis of the author's results and other author's results. The analysis is performed in section 3.6 by means of tables, for several of the studied algorithms. The comparative anlysis proves that the results obtained by the author are better than other results presented in literature: the positiong errors are similar or smaller.

3.9. References

1. [Des07] Jasmin Desai and Uf Turelli, "Evaluation Performance of Various Localization Algorithms in Wireless and Sensor Networks", 18th Annual IEEE

International Symposium on Personal, Indoor and Mobile Radio communications, 2007

2. [Che05] K.W. Chenung and H.C. So, "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements", IEEE Transactions on Signal Processing, Vol. 53, No.2, pp. 460-470, February 2005
3. [[Ari04] Jagoba Arias et. al., "GPS-less location algorithm for wireless sensor networks", Elsevier Computer Communications, vol. 30, pp. 403-409, 2004
4. [[Fin11] Andreas Fink and Helmut Beikirch., "Reliable Radio-based Human Tracking in Hazardous Environments", Wireless Congress- 2011 systems and applications, Munich, Germany, November 2011
5. [Jan11] Adel Javanmard and Andrea Montari, "Localization from Incomplete Noisy Distance Measurements", 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31 July-5 August 2011, pp. 1584-1588
6. [Rap02] Theodore S. Rappaport, "Wireless Communications- Principles and Practice", Prentice Hall Communication Engineering and Technologies Series, pp. 81-138, 2002
7. [Hol05] Holger, Karl and Andreas, Willig, Protocols and Architectures for Wireless Sensor Networks, Wiley Interscience, pp. 60-62, 111-146, 2005
8. [Rus10_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Maximum Likelihood Estimation Algorithm Evaluation for Wireless Sensor Networks", SYNASC 2010, Sept. 2010, pp. 95-98
9. [Rus10_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Target detection algorithm validation in WSN", ISETC 2010, Nov. 2010, pp. 373-376
10. [Rus11_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Evaluation of wireless sensor networks localization algorithms", IDAACS 2011, Sept. 2011, pp. 857-862
11. [Rus11_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Evaluation of some node localization algorithms", SIITME 2011, Oct. 2011, pp. 287-290
12. [Rus11_03] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Performance analysis of wireless sensor network node localization algorithm", SCVT 2011, Nov. 2011, pp. 1-5

4. WIRELESS SENSOR NETWORK LOCALIZATION ALGORITHMS - MATHEMATICAL APPROACH

The concept of modern localization algorithms refers to using second order cone programming and semidefinite programming as optimization methods for determining node positions. Semidefinite programming and second order cone programming fall into the category of convex optimization problems.

A convex optimization problem takes the form [Van09]:

$$\begin{aligned} &\text{minimize} \quad f_0(x) \\ &\text{subject to} \quad \begin{array}{l} f_i(x) \le 0,\ i = 1 \ldots m \\ a_i^T x = b_i,\ i = 1 \ldots p \end{array} \end{aligned} \tag{1}$$

In equation (1) $f_0 \ldots f_m$ are convex functions.

4.1. Second order cone programming

In second order cone programming (SOCP) a linear function is minimized over the intersection of an affine set and the product of second order (quadratic) cones. SOCPs are nonlinear and (convex) quadratic programs as special cases, but are less general than semidefinite programs (SDPs).

SOCP problems generally take the following form [Van09]:

$$\begin{aligned} &\text{minimize} \quad f^T x \\ &\text{subject to} \quad \|A_i x + b_i\|_2 \le c_i^T x + d_i,\ i = 1, \ldots, m \end{aligned} \tag{2}$$

In (2) $x \in R^n$ is the optimization variable, $A_i \in R^{n_i \times n}$ and $f \in R^{n \times n}$. A constraint of the form [Van09]:

$$\|A\ x + b\ \|_2 \le c^T x + d \tag{3}$$

is called a second order cone constraint.

The norm $\|\cdot\|_2$ is the standard Euclidian norm $\|u\|_2 = (u^T u)^{1/2}$

4.1.1. Problem Statement and Reformulation

The SOCP problem to be studied here is:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{m}\|z_i\|^2 + \sum_{j=m+1}^{n}\|z_j\|^2 \\ &\text{subject to} \quad d^2(z_i,z_j)-\tilde{d}_{ij}^2 \le \Delta \end{aligned} \qquad (4)$$

In (4) the index $i\,(\,j\,)$ refers to the anchors (sensors), while $d^2(z_i,z_j)$ is the Euclidian distance between anchors and sensors. The parameter Δ quantifies the positioning error.

In order to solve the SOCP problem the author introduces an upper bounds δ_i to each $\|\,z_i\,\|^2$, $i=1,\dots,m$. Then the SOCP problem can be written as:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{m}\delta_i \\ &\text{subject to} \quad \|\,z_i\,\|^2 \le \delta_i - C/m \\ &d^2(z_i,z_j)-\tilde{d}_{ij}^2 \le \Delta \end{aligned} \qquad (5)$$

where $C = \sum_{j=m+1}^{n}\|z_j\|^2$

This equation can be re-written as:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{m}\delta_i \\ &\text{subject to} \quad (x_i^2+y_i^2) \le \delta_i - C/m \\ &(x_i-x_j)^2+(y_i-y_j)^2 \le \tilde{d}_{ij}^2 + \Delta \end{aligned} \qquad (6)$$

The SOCP problem (6) takes the form given by equation (2) by making the following transformations. Given that the unknown variable is:

$$x = [\delta_1 \;\; \dots \;\; \delta_m \;\; x_1 \;\; y_1 \;\; \dots \;\; x_m \;\; y_m]^T \qquad (7)$$

The cost function associated with the SOCP problem is provided if:

$$f^T = [\underbrace{1 \;\; \dots \;\; 1}_{m}0 \;\; \dots \;\; 0] \text{ , so that:} \qquad (8)$$

$$f^T x = \sum_{i=1}^{m} \delta_i$$

From the first constraint associated with (6) is:

$$\begin{aligned} &(x_i^2 + y_i^2) \le \delta_i - C/m \Leftrightarrow \\ &\Leftrightarrow \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\| \le \delta_i - C/m \end{aligned} \tag{9}$$

The following SOCP variables are extracted by the author, according to the constraints from equation (2):

$$\begin{aligned} &d_i = -C/m \\ &b_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \\ &c_i = (0 \ \dots \ 0 \ \dots \ 1 \ \dots \ 0 \ \dots \ 0)^T \\ &A_i^T = \begin{pmatrix} 0 & \dots & 1 & \dots & 0 \\ 0 & \dots & 0 & \dots & 1 \end{pmatrix}, \end{aligned} \tag{10}$$

So that: $A_1^T x = \begin{pmatrix} 0 & \dots & 1 & \dots & 0 \\ 0 & \dots & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} \delta_i \\ x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$

The second constraint associated with (6) is:

$$\begin{aligned} &(x_i - x_j)^2 + (y_i - y_j)^2 \le d_{ij}^2 + \Delta \Leftrightarrow \\ &\Leftrightarrow \left\| \begin{pmatrix} x_i - x_j \\ y_i - y_j \end{pmatrix} \right\| \le d_{ij}^2 + \Delta \Leftrightarrow \\ &\Leftrightarrow \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_j \\ y_j \end{pmatrix} \right\| \le d_{ij}^2 + \Delta \end{aligned} \tag{11}$$

The following SOCP variables are obtained from (8):

$$\begin{aligned} &d_i = d_{ij}^2 + \Delta \\ &b_i = -\begin{pmatrix} x_j \\ y_j \end{pmatrix} \\ &c_i = 0 \end{aligned} \tag{12}$$

Having determined the necessary variables, the SOCP problem can now be implemented in Matlab.Introducing the upper bound δ_i is a necessary step for the Matlab implementation of SOCP. Depending of the number of anchors and sensors in the network this can be a complex, time consuming step. The next section presents the Matlab implementation of the solution developed in the previous section.

4.1.2. Matlab Implementation

In order to implement in Matlab the problem described in previous sections the author uses the SeDuMi toolbox.

SeDuMi is the abbreviation for self-dual-minimization and it is an add-on that allows users to solve optimization problems with linear, quadratic and semidefinite constraints. SeDuMi implements the self-dual embedding technique for optimization over self-dual homogeneous cones [Stu99].

According to the SeDuMi specifications, the following transforms must be obtained [Stu99]:

$$A_t = [A_t^{(1)}\ A_t^{(2)} \dots A_t^{(m)}] \tag{13}$$

$$A_t^{(i)} = -[b_i\ A_i] \tag{14}$$

$$b_t = -b \tag{15}$$

$$c_t = [c_t^{(1)}; c_t^{(2)}; \dots; c_t^{(m)}] \tag{16}$$

$$c_t^{(i)} = [d_i; c_i] \tag{17}$$

The obtained transforms can be applied to the considered problem.

4.1.2.1. Simulation Conditions

The simulation set-up consists of a 10x10 meter room where 20 sensors are deployed. The number of anchors is varied between a minimum of 3 and a maximum of 15.The obtained network is fully connected.

Three known anchor configurations were used: the random, anchor and hybrid.

It is important to study by comparison the three types of distribution for the anchor configurations in order to see the differences and to understand which offers better results.

4.1.2.2. Simulation Results

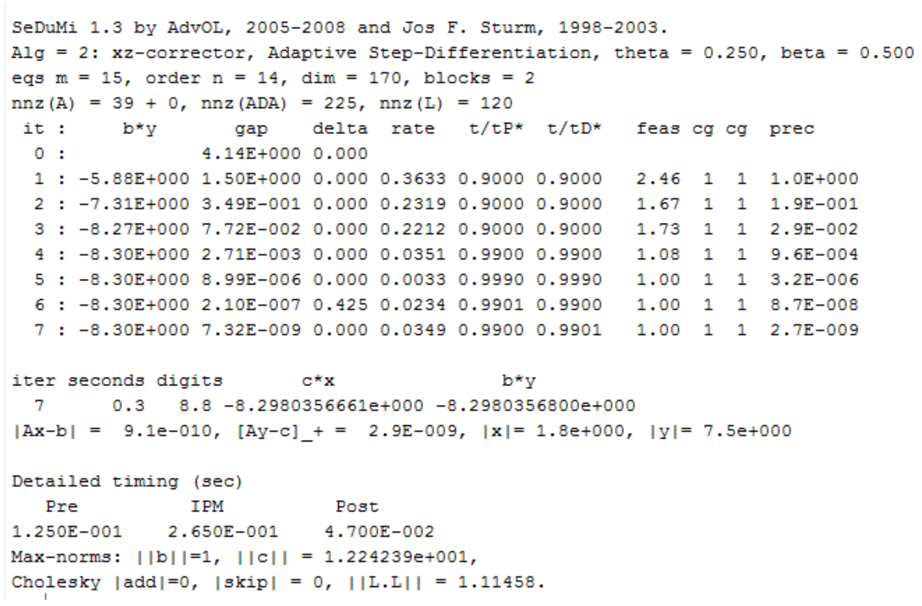
After the Matlab implementation of the considered problem, the results presented in this section were obtained.

At a first glance it can be seen that for all the considered anchor placement scenarios the average error decreases as the number of anchors increases. The highest errors are obtained for 3 anchors, while the smallest errors are obtained for 15 anchors.

In what follows, the results will be analyzed for each of the considered anchor placements.

Table 4.1 below contains the results obtained for the current simulation conditions.

Fig. 4.1 presents a SeDuMi runtime caption



```
SeDuMi 1.3 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 15, order n = 14, dim = 170, blocks = 2
nnz(A) = 39 + 0, nnz(ADA) = 225, nnz(L) = 120
 it :     b*y       gap    delta  rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            4.14E+000 0.000
  1 : -5.88E+000 1.50E+000 0.000 0.3633 0.9000 0.9000   2.46  1  1  1.0E+000
  2 : -7.31E+000 3.49E-001 0.000 0.2319 0.9000 0.9000   1.67  1  1  1.9E-001
  3 : -8.27E+000 7.72E-002 0.000 0.2212 0.9000 0.9000   1.73  1  1  2.9E-002
  4 : -8.30E+000 2.71E-003 0.000 0.0351 0.9900 0.9900   1.08  1  1  9.6E-004
  5 : -8.30E+000 8.99E-006 0.000 0.0033 0.9990 0.9990   1.00  1  1  3.2E-006
  6 : -8.30E+000 2.10E-007 0.425 0.0234 0.9901 0.9900   1.00  1  1  8.7E-008
  7 : -8.30E+000 7.32E-009 0.000 0.0349 0.9900 0.9901   1.00  1  1  2.7E-009

iter seconds digits       c*x               b*y
  7      0.3   8.8 -8.2980356661e+000 -8.2980356800e+000
|Ax-b| =  9.1e-010, [Ay-c]_+ =  2.9E-009, |x|= 1.8e+000, |y|= 7.5e+000

Detailed timing (sec)
   Pre          IPM          Post
1.250E-001    2.650E-001    4.700E-002
Max-norms: ||b||=1, ||c|| = 1.224239e+001,
Cholesky |add|=0, |skip| = 0, ||L.L|| = 1.11458.
```

Figure 4.1: SeDuMi runtime caption

A more concise approach to the results is represented by the graph in Fig. 4.2. The chart reveals that for the hybrid and uniform placements the errors tend to become stable (less variable) when using 7 to 15 anchors, while for the random placement the error becomes less variable when using 11-15 anchors. For small numbers of anchors the slope is very steep for the random placement, while for the uniform and hybrid configurations the slope is smoother (not as abrupt as for the random placement). This confirms that the random anchor placement is not very reliable and it is subject to instability when using small numbers of anchors. An average error of 4.84 meters is obtained for 3 anchors, while for 15 anchors the error is 1.25 meters. An error of 4.84 meters is quite large for a 10x10 meter room and therefore it is necessary to reduce it.

Table 4.1 SOCP simulation errors

No. of anchors	**Average error [meters]**		
	Random	**Uniform**	**Hybrid**
3	4.84 m	2.48 m	2.07 m
5	4.46 m	2.09 m	1.14 m
7	4.10 m	0.96 m	1.12 m
9	2.81 m	0.85 m	1.07 m
11	1.34 m	0.84 m	0.95 m
13	1.25 m	0.76 m	0.91 m
15	1.25 m	0.77 m	0.84 m

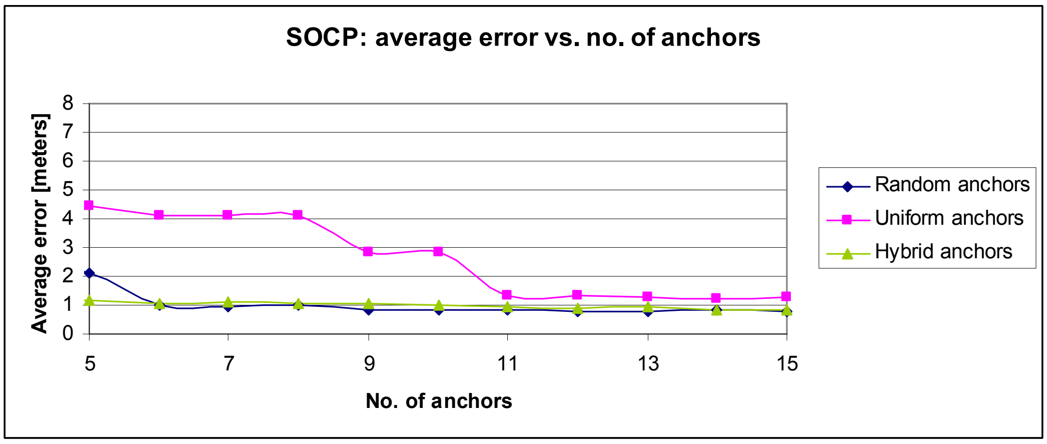


Figure 4.2: Average error vs. number of anchors

A method of reduction is using uniform anchors. For this set-up the error obtained for 3 anchors is 2.48 meters, which is approximately half than in the previous case. For 3 anchors the error is 0.77 meters and this represents again a considerable reduction when compared to the random placement.
Out of all the anchor deployment strategies, the highest errors are obtained for the random anchor configuration, while a uniform distribution is rarely encountered in practice. Therefore are situations when a certain degree of randomness for the anchor placements needs to be preserved. This can be done by using the original hybrid anchor placement. This configuration provides smaller errors than the previous placements. More precisely, the error obtained for 3 anchors is more than half when compared to the random anchors, and about 17% less than the error obtained for 3 uniform anchors. Similarly, for 15 anchors, the error is still reduced with approximately 33% when compared to the random placement, but when compared to the uniform placement the error is approximately 8% higher. Despite this, an error of 0.84 meters is more than acceptable.

4.2. Semidefinite Programming

The first step is describing the mathematical model for the localization problem. We consider a network of n distinct points in $R^d, (d \geq 1)$ and we aim to find the position of the first m points $x_l, l = 1, \ldots, m$ (sensors), given the position of $(n-m)$ points $x_i, i = m+1, \ldots, n$ (anchors) and the measured pairwise distances

δ_{lk} of sensors/sensors and sensors/anchors. The error measurement $\delta_{ij} - d_{ij}$ is considered to be arbitrary, but uniformly bounded by a given parameter Δ, that is

$|\delta_{ij} - d_{ij}| \leq \Delta$ ($d_{ij} := x_i - x_j$) is the Euclidian distance between the nodes x_i and x_j). X is the the matrix that contains the coordinates of the nodes from the network. The problem of localization has been tackled by the following algorithm based on the technique of the semidefinite programming (SDP)[Van09].

The algorithm presented by the author in this section is complex and requires a number of steps to complete it.

Input: dimension n, distance measurements d_{ij} and the bound Δ

Step 1. Solve the SDP problem:

$$\begin{aligned} &\min Tr(Q), \\ \text{subject} \quad &\text{to } |< M_{ij} Q > - d_{ij}^2 | \leq \Delta \\ &Q \pm 0 \end{aligned} \tag{18}$$

Step 2. Decompose Q by computing the best rank-d approximation, i. e.

$$Q = U_d \Sigma_d U_d^T \tag{19}$$

Step 3. Return X -the matrix of points estimates: $X = U_d \Sigma_d^{1/2}$.

Output estimated coordinates in R^d

Here, the matrix $M_{ij} = e_{ij} e_{ij}^T \in R^{nxn}$, where $e_{ij} \in R^n$ is the vector with $+1$ on the i^{th} position, -1 at the j^{th} position and 0 everywhere else; the inner product is $\langle A, B \rangle = Tr(A^T B)$ and $A \pm 0$ signifies that A is a positive semidefinite matrix.

$$e_{ij} = \begin{cases} +1, i^{th} position \\ -1, j^{th} position \\ 0, elsewhere \end{cases} \tag{20}$$

The goal of the 1st step is to look for low-rank solution Q as the Gram matrix of the position nodes $Q_0 = XX^T$ satisfies the SDP requirements and $rank(Q_0) \leq d$. This can be successful done by minimizing the $Tr(Q)$ as it expresses the sum of eigenvalues of Q.

The second step of the algorithm computes the eigendecomposition of Q and retains the first d largest eigenvalues, which is equivalent to computing the best rank approximation of Q in Frobenius norm.

The third step estimates X as the square root of Q. The main step of the above algorithm is the SDP implemented for find the matrix Q. In the following the author focuses on deriving the dual problem. To this aim the author firstly recall some basic theory regarding SDP.

The theory states that the inequality form of SDP[9] (usually refered as the primal SDP) is

$$\min_{\sum_{i=1}^{n} y_i F_i \pm F_0} \quad c^T y \tag{21}$$

where $c, y \in \mathrm{R}^n$ and F_i's are symmetric matrices. The terminology is used especially when we want to distinguish the primal problem from its dual

$$\sup_{\substack{X \pm 0 \\ tr[F_i X]=c_i, \forall i=1,\ldots,n}} \quad Tr[F_0 X] \tag{22}$$

The primal SDP is called feasible if there exits a primal feasible y, i. e. an y that satisfies all the constraints of Eq. (21) with strict inequality. The dual optimization variable is X; it is called (strictly) dual feasible if constraint inequality is (strictly) satisfied. The optimal value of (21) is usually denoted by p^* (with the convention that $p^* = \infty$ if the problem is infeasible). The optimal value of (22) is usually denoted by d^* (with the convention that $d^* = -\infty$ if the problem is infeasible). The basic facts about the dual problem are:

1. weak duality

$$p^* \geq d^* \tag{23}$$

2. strong duality: if the primal or dual problem are strictly feasible, then $p^* = d^*$

The bound (23) can sometimes be used to find a lower bound on the optimal value of a problem that is difficult to solve, since the dual problem is always convex and, in many cases, it can be solved effciently, to find d^*

4.2.1. The SDP Problem and the Derivation of its Dual

In the following the author sketches the main steps for expressing the minimization (18) in the form of the inequality SDP (21). The matrix Q subject to constraints (19) and (20) is a symmetric matrix of dimension n. It can be expressed as $Q = \sum_{1 \leq i \leq j \leq n} q_{ij} S_{ij}$ in the basis $\mathrm{B} = \{S_{ij}\}_{1 \leq i \leq j \leq n}$ of the subspace of the symmetric matrices of dimension n. Here

$$S_{ii} = \begin{cases} 1, & (i,i) - position \\ 0, & elsewhere \end{cases} \tag{24}$$

and

$$S_{ij} = \begin{cases} 1, & (i,j) \quad and \quad (j,i) - positions \\ & 0, \quad elsewhere \end{cases} \quad \text{for } i \neq j\,. \tag{25}$$

The primal optimization variable y from Eq.(21) is of the form $y = (q_{11}, q_{22}, \dots, q_{nn}, q_{12}, \dots, q_{1n}, \dots, q_{n-1,n})^T \in \mathrm{R}^d$, where $d := \frac{n(n+1)}{2}$. The trace $Tr(Q) = \sum_{i=1}^{n} q_{ii}$ can be written as $c^T y$ if $c = (\underbrace{1,\dots,1}_{n-times}, 0, \dots, 0)^T$, where the non-zero components are placed on the first n-components. The constraint $Q \pm 0$ of Eq. (18) is equivalent to

$$\sum_{1 \le i \le j \le n} q_{ij} S_{ij} \pm 0 \tag{26}$$

The second constraint Eq. (18),$| < M_{lk}, Q > - \tilde{d}_{lk}^2 \le \Delta$, expresses a double inequality for each pair $(lk), l \neq k, l = 1, \dots m, k = 2, \dots n$. Equivalently, it can be written as

$$< M_{lk}, Q > \le \Delta + \tilde{d}_{lk}^2 \tag{27}$$

$$- < M_{lk}, Q > \le -\Delta + \tilde{d}_{lk}^2 \tag{28}$$

The inner product

$$< M_{lk}, Q > = Tr(\sum_{1 \le i \le j \le n} q_{ij} e_{lk} e_{lk}^T) S_{ij} = q_{ll} + q_{kk} - 2q_{lk} \tag{29}$$

Therefore, the $2m(n-m-1)$ inequalities given by Eq.(27) become

$$\Delta - \tilde{d}_{lk}^2 + q_{ll} + q_{kk} - 2q_{lk} \ge 0 \tag{30}$$

$$\Delta + \tilde{d}_{lk}^2 - q_{ll} - q_{kk} + 2q_{lk} \ge 0 \tag{31}$$

The $2m(n-m-1)+1$ constraints (30) and (31) can be incorporated into a single inequality by combining all matrices into big block diagonal matrices which define the matrices F_0 and F_i's of Eq.(21). Each of the block matrix can be defined as containing $m(n-m-1)$ bidimensional blocks and one block matrix of dimension $n \times n$. Therefore, the matrix F_0 is of the form

$$F_0 = diag\left(F_0^{(12)}, \dots, F_0^{(1n)}, F_0^{(23)}, \dots, F_0^{(2n)}, \dots, F_0^{(mn)}, 0_{n\times n}\right) \tag{32}$$

where

$$F_0^{(lk)} = diag(\Delta - \tilde{d}_{lk}^2, \Delta + \tilde{d}_{lk}^2 \ , \ l = 1,\dots m \text{ and } k = 2,\dots n \tag{33}$$

Furthermore, the block matrices F_{lk} are of the form

$$F_{lk} = diag\left(F_{lk}^{(12)}, \dots, F_{lk}^{(1n)}, F_{lk}^{(23)}, \dots, F_{lk}^{(2n)}, \dots, F_{lk}^{(mn)}, S_{lk}\right)$$
$$l = 1, \dots, n, k = 1, \dots, n \tag{34}$$

where two different cases can be distinguished.
For $l < k$, the if $l = \dots, m$ the matrix

$$F_{lk}^{(ij)} = \begin{cases} 0_{2x2}, & \text{if} \quad (ij) \neq (lk) \\ diag(-2,2), & \text{if} \quad (ij) = (lk) \end{cases} \tag{35}$$

whereas for $l = m+1, \dots, n$ $F_{lk}^{(ij)} = 0_{2x2}$, $\forall i = 1, \dots, m, j = 1, \dots, n$.
In the particular case $l = k (= 1, \dots, n)$, we have that the matrix

$$F_{ll}^{(ij)} = \begin{cases} 0_{2x2}, & \text{if} \quad \{i,j\} \cap \{l\} = \emptyset \\ \\ diag(1,-1) & \text{if} \quad \{i,j\} \cap \{l\} \neq \emptyset \end{cases} \tag{36}$$

The derivation of the SDP dual form is presented in the following lines.

The following is an **original** result:

Proposition 1 *The SDP problem(18) can be expressed in terms of its dual as*

$$\sup \Delta Tr(X) - \sum_{1\le i \le j \le n} \delta_{ij}^2 (x_{11}^{(ij)} - x_{22}^{(ij)}) \tag{37}$$

$$X^{(ij)} = \begin{bmatrix} x_{11}^{(ij)} & x_{12}^{(ij)} \\ x_{21}^{(ij)} & x_{22}^{(ij)} \end{bmatrix} \pm 0, Y = \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \cdots & \vdots \\ y_{n1} & \cdots & y_{nn} \end{bmatrix} \pm 0 \tag{38}$$

$$Tr[F_{lk} X] = y_{lk} + y_{kl} = 0, l = m+1, \dots, n, k > l \tag{39}$$

$$-2(x_{11}^{(lk)} - x_{22}^{(lk)}) + [y_{lk} + y_{kl}] = 0, \forall l < k, l = 1, \ldots, m, k = 2, . \tag{40}$$

$$\sum_{i \neq l, i=1,\ldots,m} (x_{11}^{(il)} - x_{22}^{(il)}) + y_{ll} + \\ + \begin{cases} \sum_{j=m+1,\ldots,n} (x_{11}^{(lj)} - x_{22}^{(lj)}), if & l \leq m \\ 0, if & l > m \end{cases} = 1, \forall l = 1, \ldots, n \tag{41}$$

Proof. The standard form of the weak duality relation for semi-definite programming can be rewritten as:

$$\inf_{y \in R^n} \{c^T y \mid \sum_{i=1}^{n} y_i F_i \pm F_0\} \geq \sup_{X \pm 0} \{Tr[F_0 X] \mid \forall i = 1, \ldots, n : tr[F_i X] = c_i\} \tag{42}$$

The above inequality becomes an equality (strong duality relation) if there is a matrix $X \succ 0$ consistent with the constraints $Tr[F_i X] = c_i$.

In the following we consider X to be of the form

$$X = diag\left(X^{(12)}, \ldots, X^{(1n)}, X^{(23)}, \ldots, X^{(2n)}, \ldots, X^{(mn)}, Y\right) \tag{43}$$

where each block diagonal element is of form $X^{(ij)} = \begin{bmatrix} x_{11}^{(ij)} & x_{12}^{(ij)} \\ x_{21}^{(ij)} & x_{22}^{(ij)} \end{bmatrix}$, whereas Y is $n \times n$ -matrix.

It is easy to note that

$$Tr[F_0 X] = \sum_{1 \leq i \leq j \leq n} Tr[X^{(ij)} F_0^{(ij)}] = \sum_{1 \leq i \leq j \leq n} \Delta(x_{11}^{(ij)} + x_{22}^{(ij)}) - \delta_{ij}^2 (x_{11}^{(ij)} - x_{22}^{(ij)}) \tag{44}$$

Moreover, for $l = 1, \ldots, m, k = m+1, \ldots, n$, $l < k$, we have that

$$Tr[F_{lk} X] = Tr \sum_{1 \leq i \leq j \leq n} F_{lk}^{(ij)} X^{(ij)} + Tr[S_{lk} Y] = Tr[F_{lk}^{(lk)} X^{(lk)}] + Tr[S_{lk} Y] \\ = -2(x_{11}^{(lk)} - x_{22}^{(lk)}) + [y_{lk} + y_{kl}] \tag{45}$$

For $l = m+1, \ldots, n$, $k > l$, then

$$Tr[F_{lk} X] = y_{lk} + y_{kl} = 0 \\ = -2(x_{11}^{(lk)} - x_{22}^{(lk)}) + [y_{lk} + y_{kl}] \tag{46}$$

For $l = k \in \{1, \ldots, n\}$, then

$$Tr[F_{ll}X] = Tr \sum_{1 \le i \le j \le n} F_{ll}^{(ij)} X^{(ij)} + Tr[S_{ll}Y] =$$
$$= \sum_{1 \le i \le j \le n} Tr[F_{ll}^{(il)} X^{(ij)} + F_{ll}^{(lj)} X^{(ij)}] + Tr[S_{ll}Y] = \tag{47}$$

$$\sum_{i \ne l, i=1,\dots,m} (x_{11}^{(il)} - x_{22}^{(il)}) + y_{ll} + \left\{ \begin{array}{ll} \sum_{j=m+1,\dots,n} (x_{11}^{(lj)} - x_{22}^{(lj)}), if & l \le m \\ 0, if & l > m \end{array} \right. = 1, \forall l = 1,\dots,n \tag{48}$$

Assembling toghether Eqs. (44)-(48) in Eq.(18) it follows the dual problem associated to SDP(21).

This section presented the original algorithm developed for the adaptation of the SOCP algorithm to the problem of localization. This lagorithm is complex and requires several steps before reaching the solution. In the case of large scale networks these steps may be time consuming. The next section presents the Matlab implementation of the original algorithm developed in the current section.

4.2.2.The SDP Simulation and Obtained Results

This section presents the simulation results based on the SDP algorithm (18). The Matlab code for solving SDP uses SeDuMi (Version 1.3). In general, SeDuMi treats the primal and the dual problem associated to SDP in a symmetric way, the application particularity deciding which one to be favored over the other.

For simulating the algorithm the three simulation scenarios used before were applied. The results for a first set of data are presented in Table 4.2.

Table 4.2 SDP simulation results

No. of. anchors	**Average error [meters]**		
	Random anchor	**Uniform anchor**	**Hybrid anchor**
3	2.09	2.1	2.08
5	2.11	2.08	2.09
7	2.1	2.06	2.08
9	2.11	2.08	2.07
11	2.09	2.05	2.05
13	2.08	2.02	2.04
15	2.06	2.03	2.02

The analysis of the errors in Table 4.2 shows that the error obtained for all the three scenarios are very similar. The error vary around 2.02-2.06 meters for 3 anchors and around 2.08-2.10 meters for 15 anchors. Table 4.2 and Fig. 4.3 show that the errors suffer many variations and they decrease once the number of anchors increases. In these conditions the hybrid anchor placement seems to reduce the error introduced by the random anchor scenario.

The algorithm based on SDP(18) can be improved at least in two directions. One possible direction to follow up is to improve the size of nodes set to be studied by conceptualizing into a distributed scheme in clusters according to the anchors location. In this way the set of points is portioned and therefore the sensors can be distributed to the clusters. Hence, the SDP can be implemented separately and independently for each cluster.

Another possible direction to be explored consists of refining the solution found by specific procedures, such as the gradient descent method. The advantage of this approach is given by the use of the results obtained via the SDP described above as initial points for starting the iterations

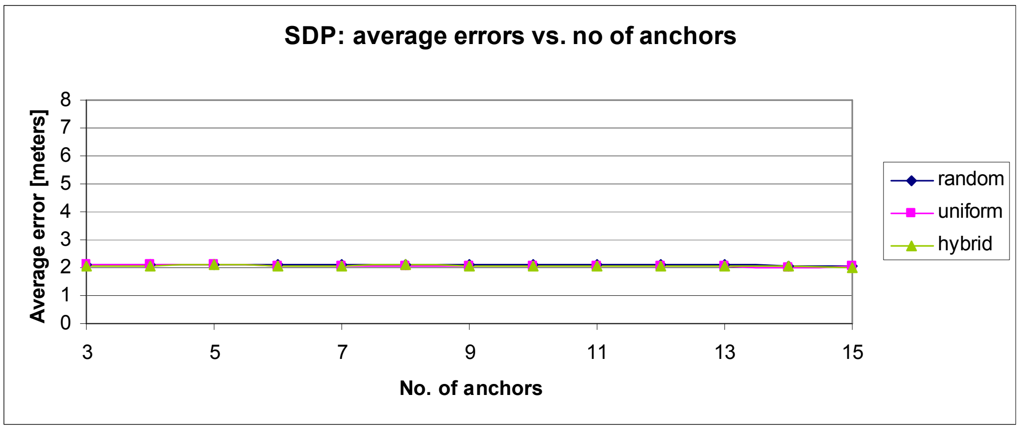


Figure 4.3a: SDP average error vs. number of anchors

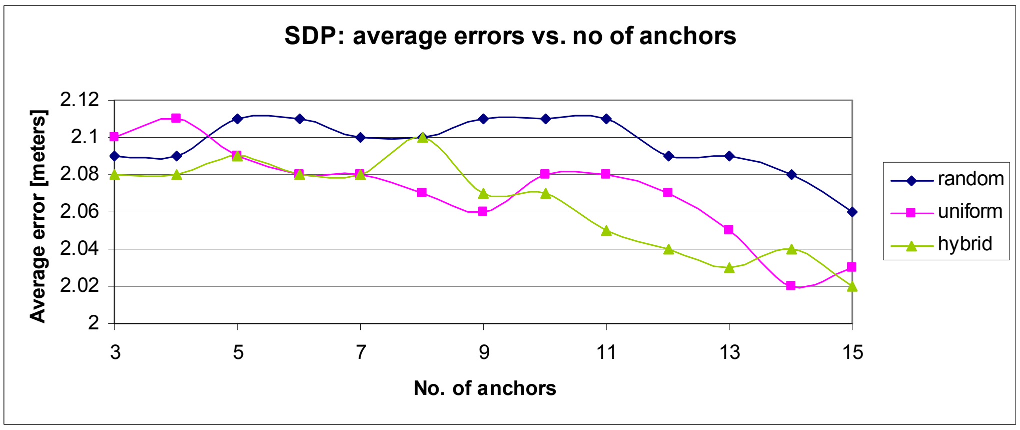


Figure 4.3b: SDP average error vs. number of anchors: close-up

4.3. Conclusions

The current chapter focused on SOCP and SDP localization of WSN nodes. The problem was given a thorough theoretical analysis, as well as a Matlab implementation. The results were analyzed by the author using three different anchor placement strategies: the classical random placement, the uniform placement, as well as novel scenario called hybrid placement. The careful analysis revealed both advantages and disadvantages of the considered scenarios, as well as the influence that the number of anchors has over the results.

The work aimed to determine an error reduction mechanism for the random and uniform placements. The goal was achieved through the hybrid anchor placement method and thus the errors obtained using this approach was smaller. When comparing the results obtained for the current algorithms with the weighted centroid localization, the iterative trilateration and the Malguki spring algorithms ([Bis04], [Sri07]), the errors are similar when using both small and high numbers of anchors, no matter which anchor placement scenario is used. However, when comparing the results with [Sri07], the current work provides better results. The algorithms with which the SOCP approach was compared are classical algorithms. This proves that the choice for classical or modern method remains the users' choice, as there are no major differences in the results obtained with either methods.

4.4. Contributions

This chapter contained a modern approach to the problem of node localization: the second order cone programming and semidefinite programming. The main contributions to this chapter are both theoretical and applicative.

From the point of view of the SOCP programming approach a theoretical contribution is the reformulating of the problem of localization for second order cone programming. This is done by introducing the upper limit δ and rearranging the newly obtained equations in such a way that the problem can be implemented into Matlab via SeDuMi toolbox. The contributions to SDP also include the original Matlab simulations and the analysis of the results, which can be included in the applicative contributions to this chapter.

The contributions regarding SDP include the obtaining of the analytical solution to the problem of semidefinite programming localization, which is a
contribution, and the original Matlab simulations. The analysis of the obtained results is another contribution to this chapter. The original simulations and the result analysis is another contribution to this chapter.

The development of the hybrid anchor placement and its verification by means of simulation is one of the author's contributions to this chapter. The original hybrid placement is both a theoretical and an applicative contribution to the current chapter.

4.5. References

1. [Hol05] Holger, Karl and Andreas, Willig, Protocols and Architectures for Wireless Sensor Networks, Wiley Interscience, pp. 60-62, 111-146, 2005
2. [Zur09] Zurawski, Richard, Networked Embedded Systsems, CRC Press, pp. 6-1-6-2, 8-1-8-21, 2009

3. [Kua07] Kuang, Xing-Hong and Shao, Hui-He, Distributed localization using mobile beacons in wireless sensor networks, The Journal of China Universites of Posts and Telecommunications, Vol. 18, Issue 4, December 2007
4. [Rus11_01] **Rusnac, Ruxandra-Ioana** and Gontean, Aurel, Performance Alalysis of Wireless Sensor Networks Node Localization Algorithms, 18th IEEE Symposium on Communications and Vehicular Technology in Benelux (SCVT 2011), Ghent, Belgium, 22-23 November 2011
5. [Des07] Desai, Jasmin and Turelli, Uf, Evaluating Performances of Various Localization Algorithms in Wireless and Sensor Networks, the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 7-12, 2007
6. [Rus11_02] **Rusnac, Ruxandra-Ioana** and Gontean, Aurel, Evaluation of Wireless sensor Networks Node Localization Algorithms, the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2011), Prague, Czech Republic, 15-17 September 2011, pp. 857-862
7. [Jav11] Javanmard, Adel and Montari, Andrea, Localization from Incomplete Noisy Distance Measurements, 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31.July-5 August 2011, pp. 1584-1588
8. [Van09] Vandenrberghe , Lieven and Boyd, Stephen, Convex Optimization, Cambridge University Press, 2009, pp. 150-188
9. [Stu99] Sturm, Jos, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optimization Methods and Software, No. 12, 1999, pp. 625-653
10. [Tse07]Tseng, Paul, Second-order cone programming relaxation of sensor network localization, Siam J. Optim, 18, 1, pp. 156-185, 2007
11. [Sri08] Srirangarajan, Seshan, Distributed Sensor Network Localization Using SOCP Relaxation, IEEE Transactions on Wireless Communcations, 7, 12,pp. 4886-4895, 2008
12. [Bao11] Baoshu, Xu, Wenyu, Qu and Wanlei, Zou, A mobile agent-based routing algorithm and some theoretical analysis, Computer Systems Science and Engineering, Vol. 26 , Issue 1 , January 2011, pp: 5-11
13. [Zho10] Zhongwen, Guo, Yuan, Feng , Lu, Hong et al., Efficient and Adaptive Transmission Algorithms for Underwater Acoustic Networks, Computer Systems Science and Engineering, Vol. 25 , Issue 6 , Special Issues, November 2010, pp. 415-425
14. [Shi11] Shirazi,Ghasem Naddafzadeh, Botros Shenouda, Michael, and Lampe, Lutz, "Second order cone programming for sensor network localization with anchor position uncertainty", 8th Workshop on Positioning Navigation and Communication (WPNC), pp. 51-55, 2011.
15. [Rus12_01] **Rusnac, Ruxandra-Ioana** and Jivulescu, Maria Anastasia, Wireless Sensor Network Node Localization via Second Order Cone Programming, in review: ISETC 2012
16. [Cox11] Cox, T and Cox, M, "Multidimensional Scaling. Monographs on Statistics and Applied Probabilities", Chapman and Hall, 2001
17. [Oh10]Oh, S et al. , "Sensor Network Localization from Local Connectivity: Performance Analysis for the MDS-map algorithm", ITW 2010
18. [Bis04] Biswas, P, "Semidefinite Programming for Adhoc Sensor Localization", IPSN 2004

19. [Sri07] Srirangarajan, S et. al, "Distributed Sensor Network Localization with Inaccurate Anchor Positions and Noisy Distance Measurements ", ISASSP, 2007
20. [Bis06] Biswas, P et. al., "A Distributed Method for Soving Semnidefinite Programming Arising for Adhoc Wireless Sensor Network Localization", Multiscale Optimization Methods and Applications, Springer Verlang, 2006

5. EXPERIMENTAL SET-UP

This chapter presents experimental results obtained using a small scale, real-life wireless sensor network. The experiments were done in two steps: first a step by step verification of Friis formula (using two nodes), after which the measurements were performed on a network made of five nodes and the studied localization algorithms were applied. The experiments were performed in an outdoor environment. The advantages of outdoor environments are the lack of reflections (multipath propagation) as well as less or no interference from other devices.

The network used for the first step is a simple network, made of a transmitter and a receiver programmed to function in API mode. The choice for this type of functioning resides in the preparatory steps for a larger network.

The experiments aim to determine the value of the duty cycle for the receiver and then, using this, the RSSI is determined. For the measurements Arduino Duemilenove boards were used by the author.

The experimental set-up is made up of a transmitter and a receiver. The measurement arrangement used for the tests is specified in [Ets06] and it typical for ZigBee based devices, such as the Xbee modules:

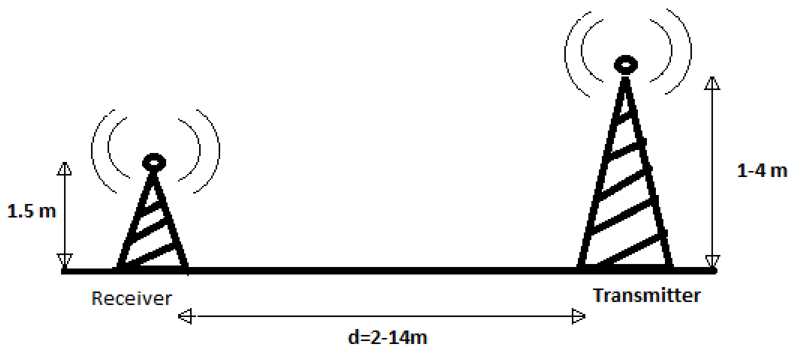


Figure 5.1: Measurement arrangement

The distance between the transmitter and the receiver is varied with 2 meters at a time, up to 14 meters.

The conversion between duty cycle and RSSI is explained in the Xbee datasheet [Xbe11]. The steps are explained in what follows.

The value of the RSSI correspond to the last received packet, as stated in [Xbe11]

$$PWM\_counts = 41 \cdot RSSI\_unsigned - 5928 \qquad (1)$$

$$RSSI\_unsigned = \frac{PWM\_counts + 5928}{41} \qquad (2)$$

For the PWM_counts variable the following calculations need to be made, keeping in mind that the PWM signal bears a total of 2400 counts [Xbe11]:

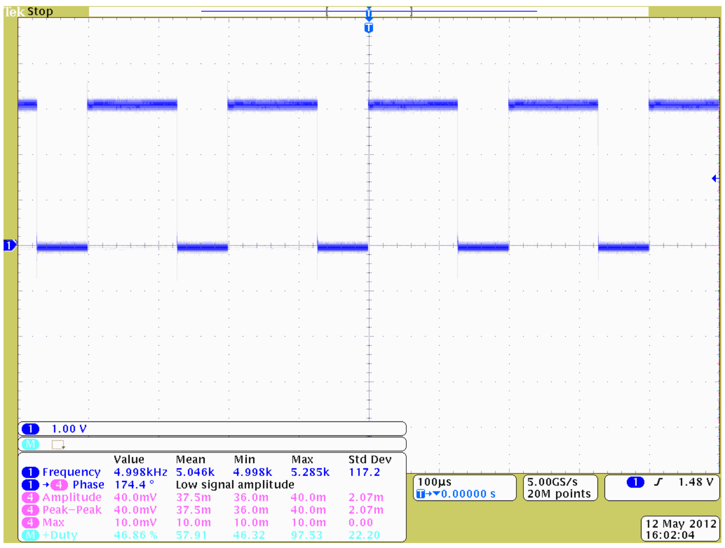
$$x\ counts = 24 * DC[\%] \tag{3}$$

By replacing (3) into (2) $RSSI\_unsigned$ is obtained:

$$RSSI\_unsigned = \frac{24 * DC[\%] + 5928}{41} \tag{4}$$

$$RSSI\_signed = 256 - RSSI\_unsigned \tag{5}$$

In Fig. 5.2 an oscilloscope capture of the PWM signal on pin 6 is presented.



	Value	Mean	Min	Max	Std Dev
1 Frequency	4.998kHz	5.046k	4.998k	5.285k	117.2
1→4 Phase	174.4 °	Low signal amplitude			
4 Amplitude	40.0mV	37.5m	36.0m	40.0m	2.07m
4 Peak−Peak	40.0mV	37.5m	36.0m	40.0m	2.07m
4 Max	10.0mV	10.0m	10.0m	10.0m	0.00
M +Duty	46.86 %	57.91	46.32	97.53	22.20

Figure 5.2: Xbee PWM signal

5.1. Xbee Transceivers

The Xbee modules are available in different versions, depending on the usage and the users' needs. For the experimental set-up the Xbee ZNet 2.5 (Series 2) modules were used. The Xbee modules are low cost devices designed to meet ZigBee/IEEE 802.15.4 standards. The modules require minimal power and provide reliable delivery of critical data between devices [Xbe11].

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other [Xbe11].

They are designed to operate indoors at distances of 100m, while outdoors they can operate at as far as 1.6 km.

The modules require a supply voltage of 3.3V. Give this and the fact that the pin spacing is not standard (as to fit in a breadboard), the modules are designed to be fitted into a socket. This socket is known as the Xbee Explorer Regulated due to the fact that it contains a voltage regulator, so that the voltage is brought down from a 5V supply voltage to 3.3 V necessary for the Xbee [Xbe11].

The modules have the following pins available [Xbe11]:

Pin no.	**Pin name**	**Direction**	**Description**
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
4	DIN/nCONFIG	Input	UART Data In
5	DIO12	Either	Digital I/O 12
6	nRESET	Input	Module reset (reset pulse must be at least 200 ns long)
7	PWM0/RSSI/DIO10	Either	PWM Output 0/ RX signal strength indicator/ Digital I/O
8	PWM/DIO11	Either	Digital I/O 11
9	nDTR/SLEEP_RQ/DIO8	Either	Pin sleep control line or digital I/O 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	nCTS/DIO7	Either	Clear to send flow control and digital I/0 7
13	ON/nSLEEP/DIO9	Output	Module status indicator and digital I/O 9
14	[reserved]	-	Must not be connected
15	Associate/DIO5	Either	Associate indicator and digital I/O 5
16	nRTS/DIO6	Either	Request to send flow control and digital I/O 6
17	AD3/DIO3	Either	Analog input 3 or digital I/O 3
18	AD2/DIO2	Either	Analog input 2 or digital I/O 2
19	AD1/DIO1	Either	Analog input 1 or digital I/O 1
20	AD0/DIO0/Commissioning button	Either	Analog input 0, Digital I/O 0, commissioning button

The following features must be taken into consideration when designing Xbee applications [Xbe11]:

- The minimum connections necessary for functioning are: VCC, GND, DOUT and DIN
- The minimum connections necessary for serial firmware updates: VCC, GND, DIN, DOUT, RTS and DTR
- Signal direction is specified for each module
- The pins that are not used must be left unconnected
- Pin 20 can be connected to a push button to support the commissioning push button functionality.

The Xbee modules are designed for serial communication. Through the serial port the Xbees can communicate with any logic and voltage compatible UART, or by means of a level translator, they can communicate with any serial device [Xbe11].

Devices that are equipped with UART can be directly connected to the pins of the Xbee module. The connection is presented in Figure 5.3 below [Xbe11].

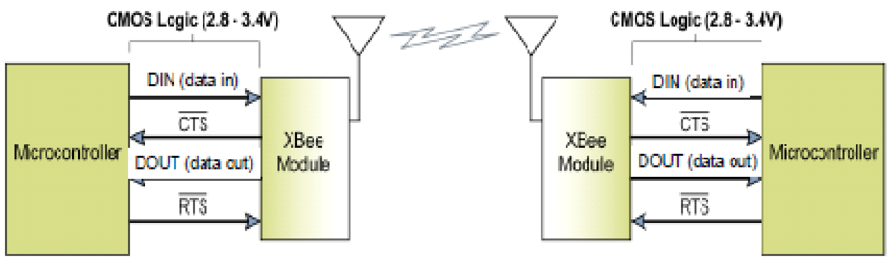


Figure 5.3: UART data flow [Xbe11]

The data to be transmitted enters the UART module through pin 3 as an asynchronous serial signal. The signal should idle high when no data is transmitted [Xbe11].

Each data byte consists of a start bit (low), 8 data bits (least significant bit comes first) and a stop bit (high). The structure of the serial data is presented in Fig. 5.4 [Xbe11].

The UART module performs parity checking, timing and all the other calculations necessary for communication. In order for two modules to communicate they have to have compatible settings: baud rate, parity, start bits, stop bits and data bits [Xbe11].

The Xbee modules support 2 modes of operation transparent and API (Application Programming Interface) serial communication [Xbe11].

When using the **transparent mode** the modules act as serial line replacement. The serial data received through the DIN pin is queued up for RF transmission. When RF data is received, the data is sent out through the DOUT pin. The module configuration parameters are configured using the AT command mode interface [Xbe11].

Transmit Data Frames (received through the DIN pin (pin 3)) include [Xbe11]:

- RF Transmit Data Frame
- Command Frame (equivalent to AT commands)
- Receive Data Frames (sent out the DOUT pin (pin 2)) include:
- RF-received data frame
- Command response
- Event notifications such as reset, associate, disassociate, etc.

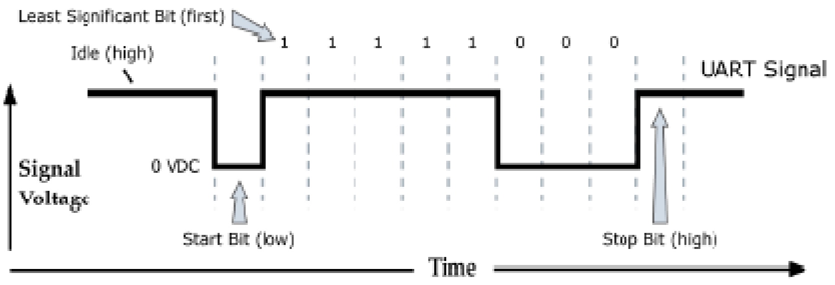


Figure 5.4: UART data package structure [Xbe11]

The API provides alternative means of configuring modules and routing data at the host application layer. A host application can send data frames to the module that contain address and payload information instead of using command mode to modify addresses. The module will send data frames to the application containing status packets; as well as source, and payload information from received data packets [Xbe11].

The API operation option facilitates many operations such as [Xbe11]:

- Transmitting data to multiple destinations without entering Command Mode
- Receive success/failure status of each transmitted RF packet
- Identify the source address of each received packet

The networks that can be created using Xbee modules are ZigBee networks. These networks are called personal area networks (PAN). Each network contains a 16-bit identifier called a PAN ID [Xbe11].

ZigBee defines three different device types – coordinator, router, and end device, elements which are typical for all wireless networks [Xbe11].

The c**oordinator** is responsible for selecting the channel and PAN ID. The coordinator starts a new PAN. Once it has started a PAN, the coordinator can allow routers and end devices to join the PAN. The coordinator can transmit and receive RF data transmissions, and it can assist in routing data through the mesh network. Coordinators are not intended to be battery-powered devices. Since the coordinator must be able to allow joins and/or route data, it should be mains powered [Xbe11].

A **router** must join a ZigBee PAN before it can operate. After joining a PAN, the router can allow other routers and end devices to join the PAN. The router can also transmit and receive RF data transmissions, and it can route data packets through the network. Since routers can allow joins and participate in routing data, routers cannot sleep and should be mains powered. Routers are optional elements in a PAN, as the network can function well without them [Xbe11].

An **end device** joins a ZigBee PAN, similar to a router. The end device, however, cannot allow other devices to join the PAN, nor can it assist in routing data through the network. An end device can transmit or receive RF data transmissions. End devices are intended to be battery powered devices. Since the end device may sleep, the router or coordinator that allows the end device to join must collect all data packets intended for the end device, and buffer them until the end device wakes and is able to receive them. The router or coordinator that allowed the end device to join and that manages RF data on behalf of the end device is known as the end device’s parent. The end device is considered a child of its parent [Xbe11].

ZigBee networks are formed when a coordinator first selects a channel and PAN ID. After the coordinator has started the PAN, routers and end devices may join the PAN. The PAN ID is selected by the coordinator when it starts the PAN.
Routers and end devices become a part of the PAN (and inherit the coordinator’s PAN ID) when they join a PAN [Xbe11].

ZigBee supports mesh routing in the network, allowing data packets to traverse multiple nodes (multiple "hops") in order to reach the destination node [Xbe11].

The main disadvantage of transparent operation is the fact that it is unicast based. Unicast transmission means that the user must specify the address of the receiving node. Unicast transmission, in transparent mode is much faster that broadcast (one node transmitting to multiple nodes). Broadcast in transparent mode introduces delays in the transmission [Xbe11].

API operation is an alternative to transparent operation. The frame-based API extends the level to which a host application can interact with the networking capabilities of the module. When in API mode, all data entering and leaving the module is contained in frames that define operations or events within the module[Xbe11].This allows ZigBee nodes to be spread out over a large region, and still support communications amongst all devices in the network [Xbe11].

All devices in a ZigBee network receive a 16-bit address (network address) when they join a PAN. The 16-bit address of the coordinator is always 0 [Xbe11].

It is possible to measure the received signal strength on a device using the DB command. DB returns the RSSI value (measured in –dBm) of the last received packet [Xbe11].

The DB value can be determined in hardware using the RSSI/PWM module pin (pin 6). If the RSSI PWM functionality is enabled (P0 command), when the module receives data, the RSSI PWM is set to a value based on the RSSI of the received packet [Xbe11].

5.2. APIC Development Board Measurement

The APIC development board was developed in the Faculty of Electronics and Telecoomunications (UPT). The board is based on a PIC 18F455x microcontroller. These are 8 bit advanced microcontrollers from Microchip. One feature of these microcontrollers is its capture and compare module, CCP. This module, according to its datasheet, contains a 16-bit register which can operate as a16-bit capture register, as a 8-bit compare register or as a 10-bit PWM master/slave Duty Cycle register. The CCP modules are identical in operation, with the exception of the operation of the special event trigger. Each CCP module has 3 registers. Multiple CCP modules may exist on a single device.

In order to measure the PWM signal it was necessary to use the CCP module in capture mode. The software functions as follows:

1. the capture module is set to capture the rising edge of the incoming signal
2. timer 1 is configured
3. the CCP interrupt is enabled
4. when the desired edge of the signal appears the value of the timer is saved
5. the capture module is configured to capture the falling edge of the signal
6. when the falling edge appears the value of the timer is saved again
7. the pulse width is obtained as difference of the timer value read at step 6 and the timer value read at step 4
8. using a separate function the period of the signal is measured

The distance between the sender and the receiver was varied with 2 meters at a time, up to a maximum of 14 meters. In Table 5.1 the values of the duty cycle, the RSSI and the received power are represented, for a first set of data.

The values of the duty cycle decrease once the distance between the transmitter and the receiver increases. The same is valid for the received power, which decreases with distance. However, it can be observed that for different distances between the two nodes the duty cycle and RSSI values are repetitive. This makes the results unsuitable for further processing. To make things more clear, a representation of the measured RSSI and the theoretical RSSI is made in Fig. 5.7. The figure shows almost no variation for RSSI, until a distance of 12 meters between sender and receiver. At this point a peak appears. What is even more obvious from the graph is the large difference between the measured and the theoretical values of RSSI. The difference is 40 dBm. This is a large value and proves once again that the results obtained using APIC boards are not feasible. Therefore other methods must be used.

The C code used to measure the PWM signal is presented in Fig. 5.5.

Table 5.1 APIC measurement results

Distance [meters]	**Duty Cycle**	**RSSI$_M$ [dBm]**
2	0.53	-80.09
4	0.41	-87.12
6	0.41	-87.12
8	0.41	-87.12
10	0.41	-87.12
12	0.59	-76.58
14	0.30	-93.56

5.2.1. Presentation of Arduino Duemilenove

The Arduino Duemilanove ("2009") is a microcontroller board based on the ATmega168 or ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button [Ard11].

The Arduino Duemilanove can be powered via the USB connection or with an external power supply. External power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. The recommended range is 7 to 12 volts [Ard11

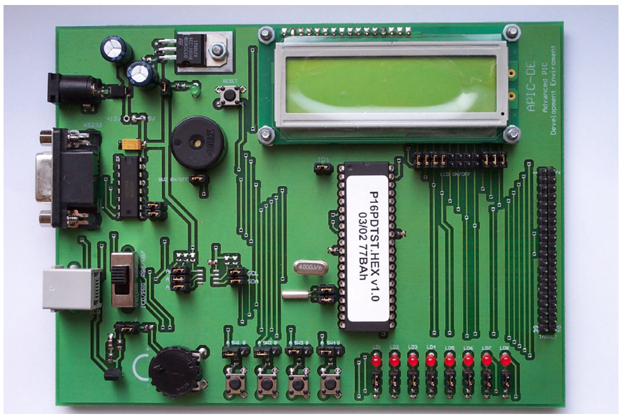


Figure 5.6: APIC development board

```
/* Author: Ruxandra Rusnac */
#include <p18f452.h>
#include <timers.h>
#include <stdlib.h>
#include <capture.h>
#pragma config OSC = XT
#pragma config WDT = OFF
#pragma config LVP  = OFF
#pragma config DEBUG = OFF
unsigned int ov_cnt, temp;
unsigned short long period;

void high_ISR(void);
#pragma code high_vector
void high_interrupt(void)
{         _asm
          goto high_ISR
          _endasm}

#pragma interrupt high_ISR
void high_ISR(void)
{         if (PIR1bits.TMR1IF)
                    PIR1bits.TMR1IF=0;
                    ov_cnt++;}

void main(void)
{         unsigned int temp1,rise,fall,highTime;
          int edge=0;
          ov_cnt=0;
          INTCONbits.GIE=0;
          RCONbits.IPEN=1;
          PIR1bits.TMR1IF=0;
          IPR1bits.TMR1IP=1;
          TRISCbits.TRISC2=1;
          OpenTimer1(TIMER_INT_ON & T1_16BIT_RW & T1_PS_1_1 &
                              T1_OSC1EN_OFF & T1_SYNC_EXT_OFF & T1_SOURCE_INT);
          OpenTimer3(TIMER_INT_OFF & T3_16BIT_RW & T3_PS_1_1 &
                              T3_SOURCE_INT & T3_PS_1_1 & T3_SYNC_EXT_ON &
                              T1_SOURCE_CCP);
          OpenCapture1(CAPTURE_INT_OFF & C1_EVERY_RISE_EDGE);
          PIE1bits.CCP1IE=0;
          PIR1bits.CCP1IF=0;
while(1){           if (PIR1bits.CCP1IF==1)
          {                   if (edge==0)                //rising edge
                    {         rise=CCPR1;
                              CCP1CON=0x04;
                              edge=1;
                              PIR1bits.CCP1IF=0;}
                    else
                    {         fall=CCPR1;
                              highTime=fall-rise;
                              CCP1CON=0x05;
                              edge=0;
                              PIR1bits.CCP1IF=0;
                    }}}}
```

Figure 5.5: APIC PWM measurement code

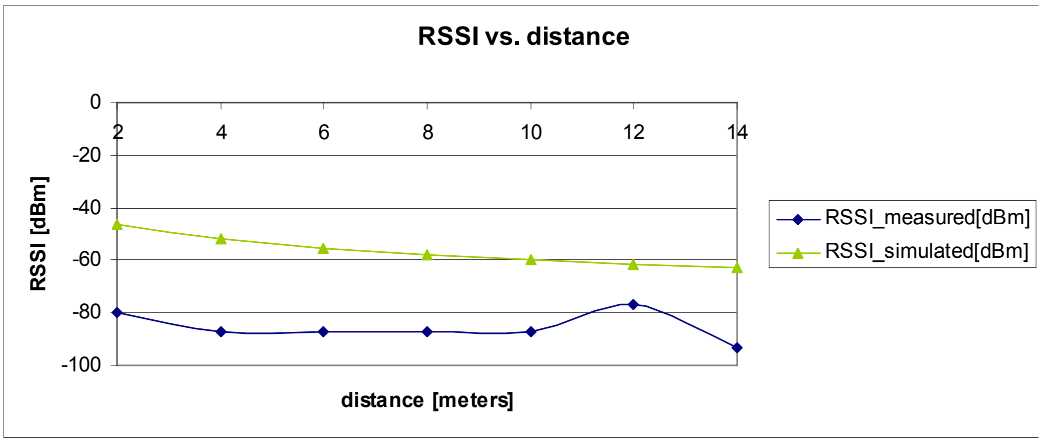


Figure 5.7: APIC RSSI vs. distance

5.3. Arduino Measurements

The ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader); the ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the EEPROM library); the ATmega328 has 2 KB of SRAM and 1 KB of EEPROM [Ard11].

The Arduino Duemilanove has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega168 and ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with Windows version of the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1) [Ard11].

A SoftwareSerial library allows for serial communication on any of the Duemilanove's digital pins [Ard11].

The ATmega168 and ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus [Ard11].

In order for the Aduino board to function with Xbee modules it is necessary to use an Xbee Shield. The Xbee shiled works as an extension board for the Arduino. This is mounted on the considered board and it offers the possibility to control every aspect of the Xbee module with the Aduino Duemilenove board [Ard11].

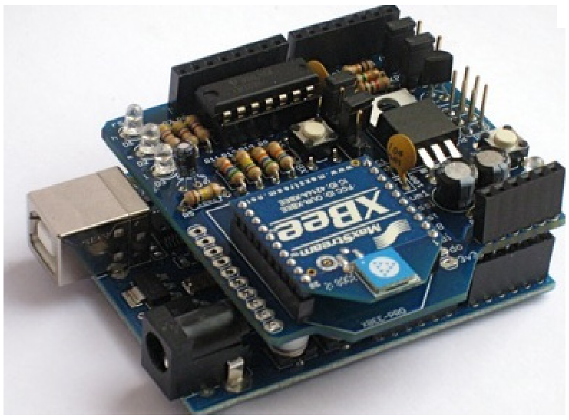


Figure 5.8: Arduino Duemilenove with Xbee Shield

5.3.1. Measurement Set-up and Results

The measurement set-up is similar to the one presented in Fig.5.1, except for the fact that the Arduino Duemilenove is connected to the PC in order to read the results. This is necessary because this board in not equipped with and LCD, as the APIC board is.

The Arduino board can be programmed using the Arduino software. This software is based on the Processing language, which is situated between C and Java. The ATmega168 or ATmega328 on the Arduino Duemilanove comes preburned with a bootloader that allows code uploading without the use of an external hardware programmer. It communicates using the original STK500 protocol [Ard11].

To measure the PWM signal duty cycle the Arduino software built in function *pulseIn()* was used. This function reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. It returns the length of the pulse in microseconds. Gives up and returns 0 if no pulse starts within a specified time out. [Ard11].

The syntax of *pulseIn* is [Ard11]:

pulseIn(pin, value)
pulseIn(pin, value, timeout)

The parameters of the function are:

- pin: the number of the pin on which you want to read the pulse. It is an integer.
- value: type of pulse to read: either HIGH or LOW. The value is an integer
- timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. This is unsigned longThe *pulseIn* function returns the length of the pulse (in microseconds) or 0 if no pulse started before the timeout [Ard11].

The measurement procedure is the same as before.

In order to obtain the values for the received power the following calculations are necessary:

$$RSSI[dBm] = 10\log\frac{P_{Rx}}{P_{Tx}} \tag{6}$$

$$\frac{RSSI[dBm]}{10} = \log P_{Rx} - \log P_{Tx} \tag{7}$$

The transmit power, P_{Tx} is 1 mW, therefore the value for the received power can be written as:

$$P_{Rx} = 10^{\frac{RSSI-30}{10}} \tag{8}$$

Equation (8) will be used for calculating the values for the received power.

In Table 5.2 the measurement results are presented.

The duty cycle decreases with distance: for 2 meters the value is 0.915, while for 14 meters the value is 0.68. Due to this fact RSSI becomes smaller with distance. A maximum of -57.85 is measured for 2 meters. The minim RSSI is -71.60 at 14 meters. Accordingly the received power decreases with distance. There are no repetitive values of the duty cycle, RSSI or received power, thus making the data suitable for further processing.

The representation of the theoretical (simulated) and measured RSSI is in Fig. 5.9.

The error between the simulated and the measured RSSI is small. It is approximately 10-15 dBm, which is considerably smaller than in the APIC situation.

Table 5.2 Arduino measurement results, set I

Distance [meters]	**Duty Cycle**	**$RSSI_M$ [dBm]**	**P_{Rx}[Watt]**
2	0.91	-57.85	$1.63\ 10^{-9}$
4	0.96	-55.21	$3.00\ 10^{-9}$
6	0.85	-61.36	$7.30\ 10^{-10}$
8	0.81	-64	$3.98\ 10^{-10}$
10	0.83	-62.82	$5.21\ 10^{-10}$
12	0.84	-61.95	$6.38\ 10^{-10}$
14	0.68	-71.60	$6.90\ 10^{-11}$

An illustration of DC variation can be found in Fig. 5.10. The slope of the duty cycle is not smooth. Variations occur at 4 and 12 meters, where peaks can be observed. However, the duty cycle decreases once the distance between the transmitter and the receiver increases. This is expected from the theoretical aspects of the problem.

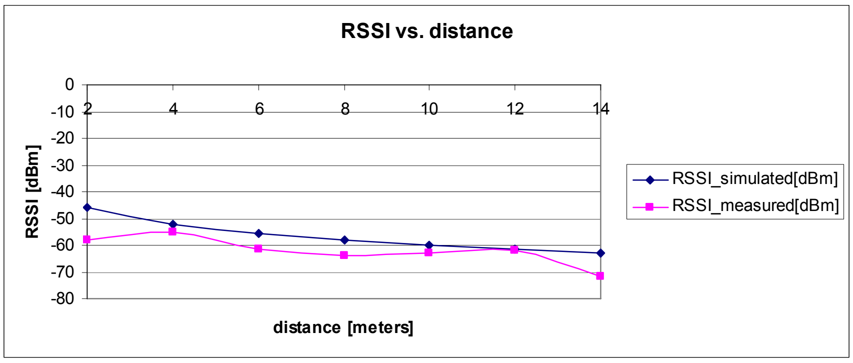


Figure 5.9: Arduino: RSSI vs. distance, set I

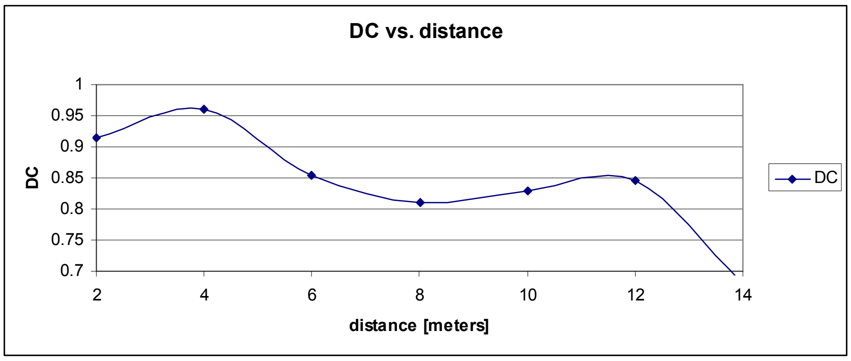


Figure 5.10: Arduino: DC vs. distance, set I

In Fig. 5.11 the variation of the measured received power against distance is presented.

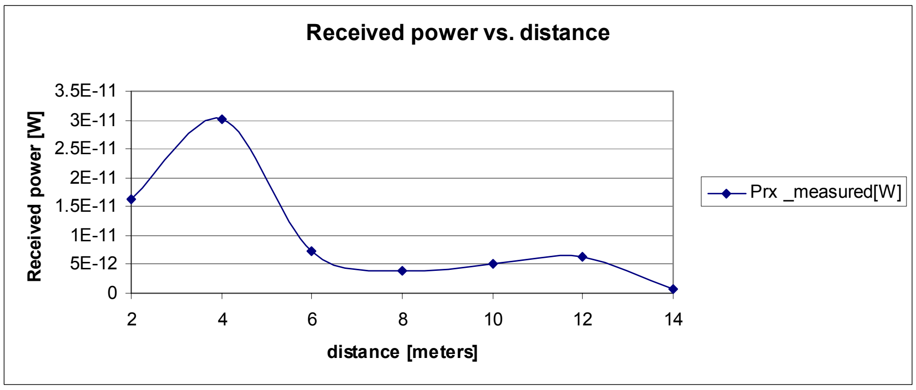


Figure 5.11: Arduino: Received power vs. distance, set I

The slope of the received power respects the slope of the duty cycle as well as the slope of the RSSI.

A second set of results is presented in Table 5.3.

The table reveals that the decrease of duty cycle with distance is maintained. In accordance the RSSI decreases with distance and the received power becomes smaller as the distance between the transmitter and the receiver increases. All these aspects are expected from theory. For confirmation, in Fig. 5.15 the variation of RSSI with distance is presented and in Fig. 5.16 the variation of DC with distance is presented.

In Fig. 5.14 the received power with distance is presented for the measured received power. Variation for curve for the received power, RSSI and duty cyle are similar.

A third set of results is presented in Table 5.4. In this situation the slopes of the measured and the theoretical RSSI are closer to each other than in the previous situations. The error between measurement and theory is reduced to 0-5 dBm for distances of 2-14 meters. The duty cycle presents a peak at 4 meters but all in all it decreases with distance, as it should.

Table 5.3 Arduino measurement results, set II

Distance [meters]	**Duty Cycle**	**$RSSI_M$ [dBm]**	**P_{Rx}[Watt]**
2	0.96	-55.21	$3.00\ 10^{-9}$
4	0.93	-56.97	$2.00\ 10^{-9}$
6	0.83	-62.82	$5.21\ 10^{-10}$
8	0.77	-66.04	$2.48\ 10^{-10}$
10	0.79	-65.17	$3.04\ 10^{-10}$
12	0.85	-61.65	$6.82\ 10^{-10}$
14	0.70	-70.14	$9.66\ 10^{-11}$

Table 5.4 Arduino measurement results, set III

Distance [meters]	**Duty Cycle**	**$RSSI_M$ [dBm]**	**P_{Rx}[Watt]**
2	0.92	-57.56	$1.75\ 10^{-9}$
4	0.96	-55.21	$3.00\ 10^{-9}$
6	0.86	-61.07	$7.81\ 10^{-10}$
8	0.87	-60.19	$9.56\ 10^{-10}$
10	0.87	-60.19	$9.56\ 10^{-10}$
12	0.82	-63.12	$4.87\ 10^{-10}$
14	0.72	-68.97	$1.26\ 10^{-10}$

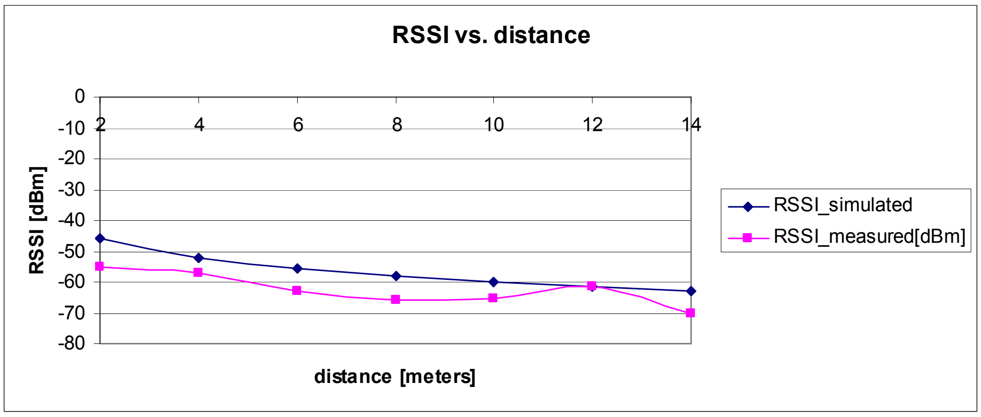


Figure 5.12: Arduino: Received power vs. distance, set II

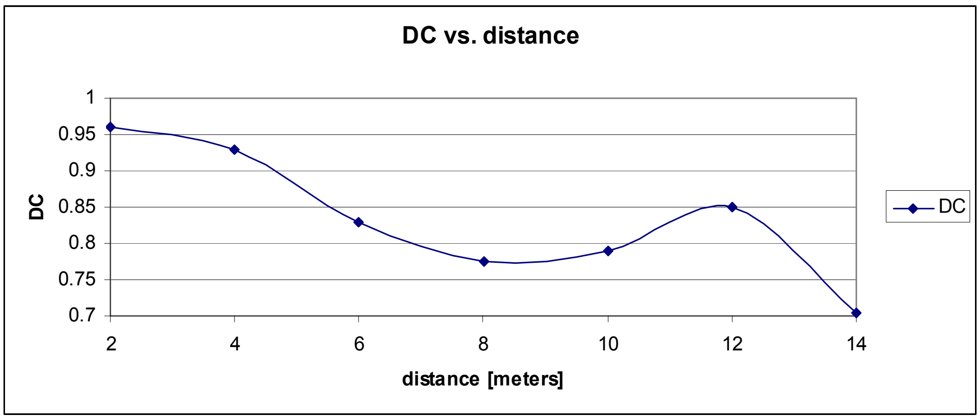


Figure 5.13: Arduino: Duty cycle vs. distance, set II

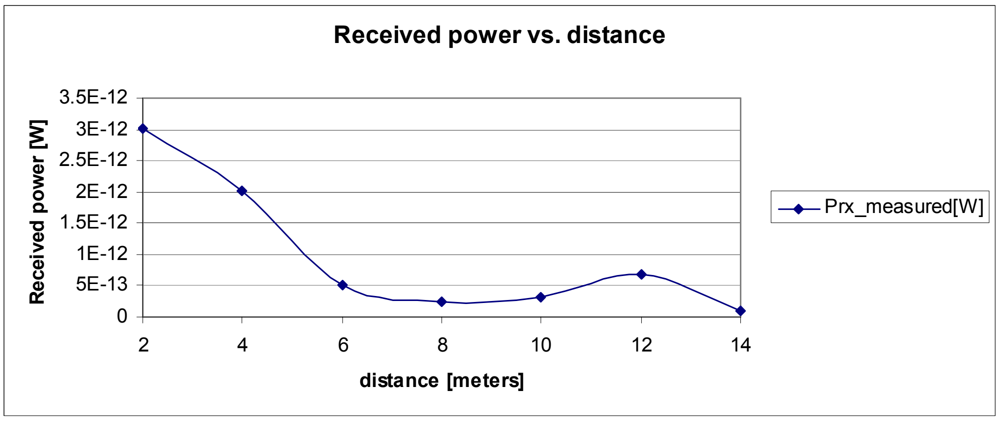


Figure 5.14: Arduino: Received power vs. distance, set II

In Fig. 5.17 the received power variation with distance is presented.

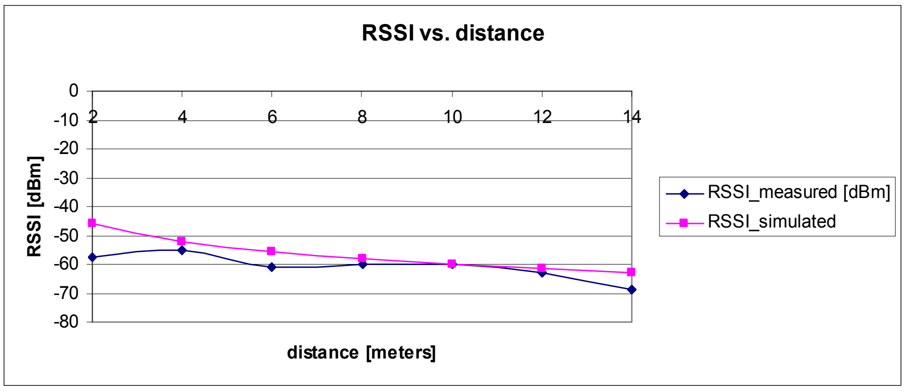


Figure 5.15: Arduino: Received power vs. distance, set III

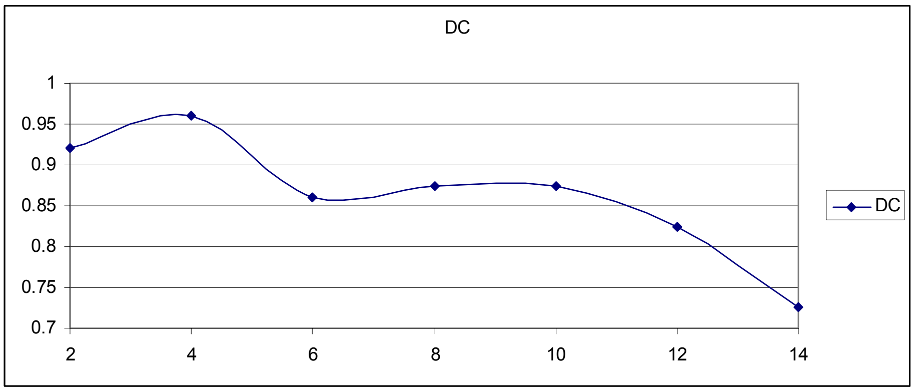


Figure 5.16: Duty cycle vs. distance, set III

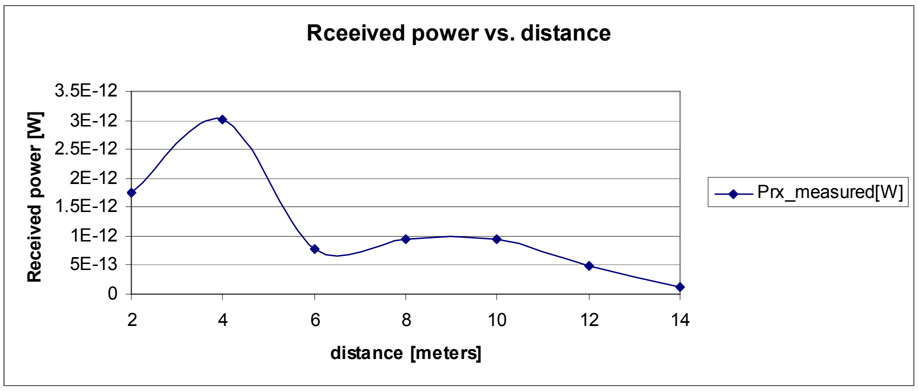


Figure 5.17: Arduino: Received power vs. distance, set III

The slope for the measured received power corresponds to the slope of the duty cycle and to the slope of RSSI. Even though at a distance 4 meters between the transmitter and the receiver there is a peak, the decrease of the received power with distance can be observed in Fig. 5.17.

In Table 5.5 the results for a fourth set of data are presented.The duty cycle has a maximum of 0.92 and a minimum of 0.68 for 14 meters. In between the two extremes the duty cycle values decrease. This is as expected from theory. This decrease in duty cycle results in a decrease of RSSI with distance and as well as in a decrease of the received power. The RSSI variation with distance and the duty cycle variation with distance is presented in Fig. 5.18 an 5.19.

Table 5.5 Arduino measurement results, set IV

Distance [meters]	**Duty Cycle**	**$RSSI_M$ [dBm]**	**P_{Rx}[Watt]**
2	0.92	-57.56	$1.75\ 10^{-9}$
4	0.63	-74.53	$3.51\ 10^{-9}$
6	0.86	-61.07	$7.81\ 10^{-10}$
8	0.85	-61.65	$6.82\ 10^{-10}$
10	0.84	-61.95	$6.38\ 10^{-10}$
12	0.85	-61.36	$7.30\ 10^{-10}$
14	0.68	-71.60	$6.90\ 10^{-11}$

In this particular situation the measured RSSI varies slightly different from the simulated RSSI, especially for 2-6 meters. Due to this fact a larger error exists at these distances: 30 dBm, while for the rest of the measurement range the error is reduced to almost zero. The duty cycle variation presents a variation at 4 meters, where the duty cycle decreases dramatically. Then it increases and it becomes more stable. All in all the duty cycle tends to decrease with distance.

In Fig. 5.20 the received power variation is presented. The slopes for the received power, RSSI and duty cycle are similar.

Due to the fact that the results for the four data sets are very similar the author does not consider it necessary to present the rest of the results that were obtained.

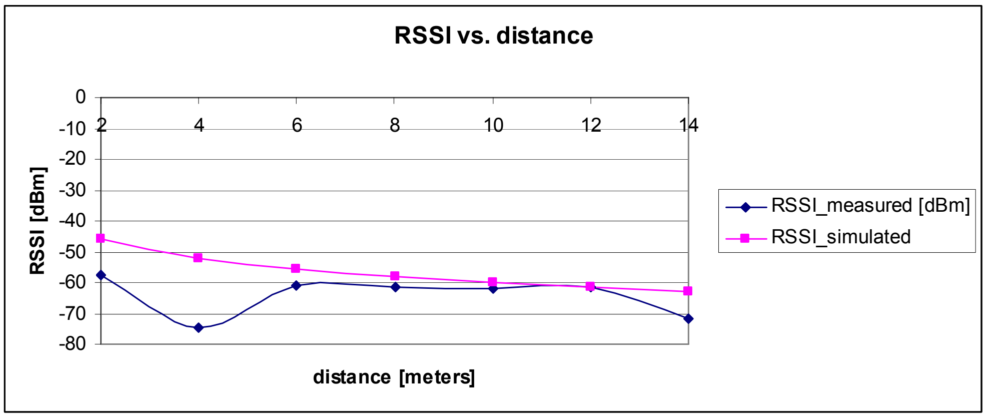


Figure 5.18: Arduino: RSSI vs. distance, set IV

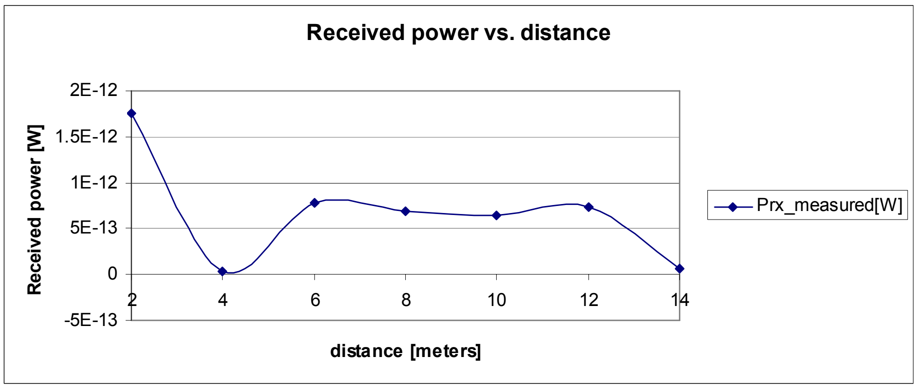


Figure 5.19: Arduino: Duty cycle vs. distance, set IV

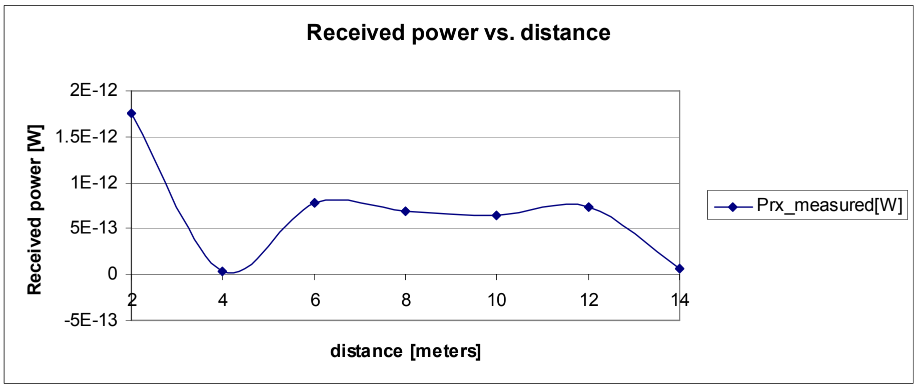


Figure 5.20: Arduino: Received power vs. distance, set III

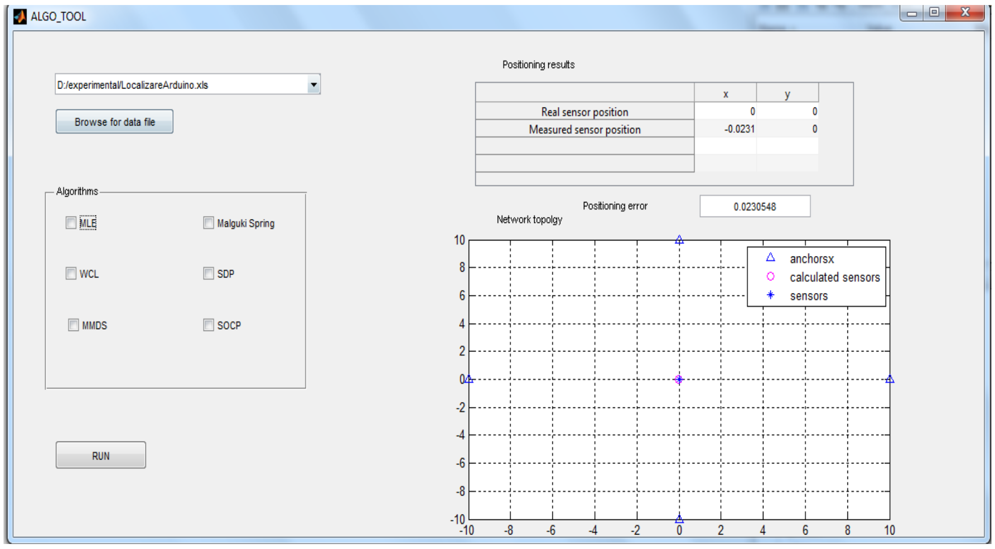
5.4. ALGO_TOOL

In order to offer an easier and faster way of applying the algorithms to the network the author designed in Matlab a graphical user interface (GUI) called 'ALGO_TOOL'. The advantage of this tool is the compact manner in which it processes the data and in which it displays the results.

ALGO_TOOL is based on the idea that the information necessary for the algorithms is stored in an Excel file. The structure of this file must be entirely respected and the information should be written on the columns designed for the tool. The user has the possibility to browse for the file and after making a choice the filename is displayed. This is meant to be a confirmation of the right choice for the file.

Next the user can select from the panel 'Algorithms' one of the six algorithms to be run. Choosing an algorithm disables the checkbox for the remaining algorithms, so that conflicts are avoided.

After choosing the file and the algorithm, the user must press the 'RUN' button. This starts the calculations. When the calculations are done the results are displayed in two forms. The first form is a table containing the real and the measured sensor positions and the obtained positioning error. The second way is by means of a graph: the network topology. The graph presents the anchors, the measured and the calculated sensors on the same axes.


ALGO_TOOL

D:/experimental/LocalizareArduino.xls

Positioning results

	x	y
Real sensor position	0	0
Measured sensor position	-0.0231	0

Positioning error: 0.0230548

Figure 5.21: ALGO_TOOL

5.5. Localization Results

In the next section a real life localization scenario is presented. The network is composed of 4 anchors and one sensor and it is placed in free space. For the communication between nodes the Arduino Duemilenove and Xbee nodes are used.

The communication is in broadcast mode. Using the values of RSSI the distance between the network nodes is determined. The anchor positions are known. Using all this information the algorithms simulated in chapter 3 are applied and the position of the sensor is determined. The surface on which the network spreads is a 10 x10 meters, with the sensor being placed in the center of this surface. In what follows the results obtained for each board are presented from two points of view: the localization error and the sensor positions.

The sensor in placed in the center of the network, which corresponds to the $S(0,0)$, while the anchors are placed as follows: $A_1(0,10)$, $A_2(10,0)$, $A_3(0,-10)$ and $A_4(-10,0)$.

The algorithms applied to the network are: maximum likelihood estimation, weighted centroid localization, Malguki spring and modified multidimensional scaling, second order cone programming and semidefinite programming.

5.5.1. Maximum Likelihood Estimation

Maximum likelihood estimation is the simplest and fastest algorithm out of all the implemented algorithms. It was applied for two sets of measurements for the considered network.

In Table 5.6 the real and the measured sensor position are presented, together with the positioning error. The results are presented for a first round of measurements.

Table 5.6 MLE positioning, set I

	x	y
$S_{measured}$	-0.90	-0.89
$d(S_{real}, S_{measured})$ [meters]	1.27	

From Table 5.6 the positioning error can be determined. For the current measurements the positioning error is 1.27 meters. This value can be analyzed depending on the application where the network will be used: it can be critical or not.

In Fig. 5.22 the network is presented, together with the real and the calculated anchor positions. From the figure the difference between the measured and the real position is easy to observe.

A comparative analysis of Table 5.6 and 5.7 reveals that the results obtained for the second set of measurements are better than the results obtained for the previous set. The positioning error is 0.02 meters, opposed to 1.27. The error for this set of measurements is considerably reduced that before

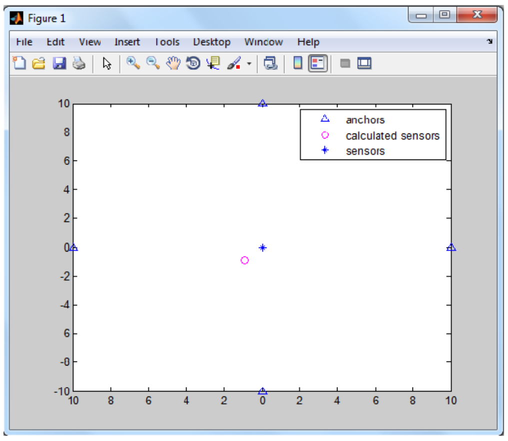


Figure 5.22 MLE network, set I

In Table 5.7 the results for the second set of measurements are presented.

Table 5.7 MLE positioning, set II

	x	y
$S_{measured}$	-0.02	0
$d(S_{real}, S_{measured})$ [meters]	0.02	

5.5.2. Weighted Centroid Localization

This algorithm is more complex then the MLE and thus slightly slower. The results for this algorithm are presented in this section for the networks considered before.

The first set of results if presented in Table 5.8.

The positioning error obtained for the current set is very high compared to the previous cases. This result is expected from simulation, when WCL proved to offer worse results than the MLE.

Table 5.8 WCL positioning, set I

	x	**y**
$S_{measured}$	-1.83	-3.44
$d(S_{real}, S_{measured})$ [meters]	3.90	

Table 5.9 presents WCL results for a second set of data.

The positioning error obtained here is lower than for the first data set. However it is still higher than the error obtained by applying MLE for the same data set. Still the error is in acceptable limits.

Table 5.9 WCL positioning, set II

	x	**y**
$S_{measured}$	-0.67	0
$d(S_{real}, S_{measured})$ [meters]	0.67	

5.5.3. Malguki Spring

The increasing complexity of this algorithm requires more computational time. However, the results provided by this algorithm are good.

A first set of results is presented in Table 5.10.

The positioning error obtained in this case is high: 2.19 meters. For this case the algorithms proves to be unreliable. Compared to the other algorithms the error is higher than the MLE error but lower that the WCL error.

Table 5.10 Malguki positioning, set I

	x	**y**
$S_{measured}$	-1.55	-1.55
$d(S_{real}, S_{measured})$ [meters]	2.19	

The results obtained for the second set of data are very similar to the first set and therefore they will not be presented once again in this section. The reason for the similarity resides in the small differences between the RSSI values for the two analyzed data sets as well as in the use of optimization for calculations and the starting point to which the algorithm must converge to.

5.5.4. Modified Multidimensional Scaling

The results for MMDS are presented in this section from the point of view of two data sets. Table 5.11 contains the first results.

The positioning error provided by MMDS is the smallest of all the errors obtained up to this point. This proves that, even though the algorithm is complex and slow the results are reliable due to the reduced positioning error.

The results for a second data set are very similar to the previous results and therefore the network topology is not presented once again. As for Malguki spring, the reason for the similarity resides in the small differences between the RSSI values for the two analyzed data sets as well as in the use of optimization for calculations and the starting point to which the algorithm must converge to.

Table 5.11 MMDS positioning, set I

	x	**y**
$S_{measured}$	0.01	0.01
$d(S_{real}, S_{measured})$ [meters]	0.01	

5.5.5. Semidefinite Programming

Table 5.12 presents the localization results obtained for semidefinite programming. The results are for the first data set that was analyzed.

Table 5.12 SDP positioning, set I

	x	**y**
$S_{measured}$	-0.48	-0.48
$d(S_{real}, S_{measured})$ [meters]	0.67	

The positioning error obtained after using this algorithm is 0.67 meters, which represents an error small enough to make this algorithm suitable for real life applications that do not demand very high precision (life threatening).

The positioning results for the second data set that the author analyzed are in Table 5.13.

Table 5.13 SDP positioning, set I

	x	**y**
$S_{measured}$	0.28	0.28
$d(S_{real}, S_{measured})$ [meters]	0.39	

The positioning error of 0.39 meters is smaller than the positioning error that was obtained for the first set of data. This value recommends the algorithm for use in real life applications. Still the recommendation holds that the algorithm should not be used in life depending applications.

5.5.6. Second Order Cone Programming

The last positioning algorithm studied by the author is SOCP. The algorithm was applied to two sets of data. The performance was analyzed from the point of view of the positioning error.

Table 5.14 presents the results obtained for the first set of data that was analyzed.

The positioning error obtained after this algorithm is once again very small. This value is comparable to the results obtained after applying MMDS. Such errors make the algorithm suitable for real life applications. The network layout is in Fig. 5.23. The layout confirms the small positioning error: the real and the measured anchor positions are identical.

Table 5.14 SOCP positioning, set I

	x	**y**
$S_{measured}$	0.01	0.01
$d(S_{real}, S_{measured})$ [meters]	0.01	

The results for the last data set that was analyzed are presented in Table 5.15.

Table 5.15 SOCP positioning, set II

	x	**y**
$S_{measured}$	0	0
$d(S_{real}, S_{measured})$ [meters]	0	

The measured sensor position coincides with the real sensor position. Therefore the positioning error is null

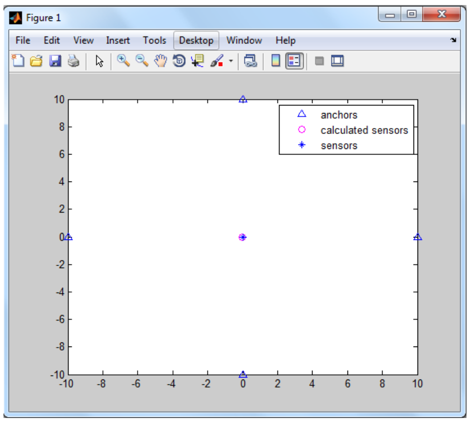


Figure 5.23: SOCP network, set I

5.6. Averaged RSSI

As stated in the introduction, the RSSI corresponds to the last received packet. Even though for the results presented in this section the error between theory and measurement was reduced, the author finds it useful to study the effect of averaging several consecutive RSSI readings. For this, the author averaged 100 consecutive readings. Averaging should, at theoretical level, increase the accuracy of the results. In what follows a selection of the results is presented

In Table 5.16 the first set of averaged RSSI is presented.

Table 5.16 Arduino average RSSI, set I

Distance [meters]	**$RSSI_{averaged}$ [dBm]**	**P_{Rx}[Watt]**
2	-56.39	$2.29\ 10^{-9}$
4	-58.14	$1.53\ 10^{-9}$
6	-60.48	$8.93\ 10^{-10}$
8	-62.24	$5.96\ 10^{-10}$
10	-61.65	$6.82\ 10^{-10}$
12	-63.41	$4.55\ 10^{-10}$
14	-69.85	$1.03\ 10^{-10}$

Fig. 5.24 represents the variations of the averaged RSSI and the theoretical RSSI.

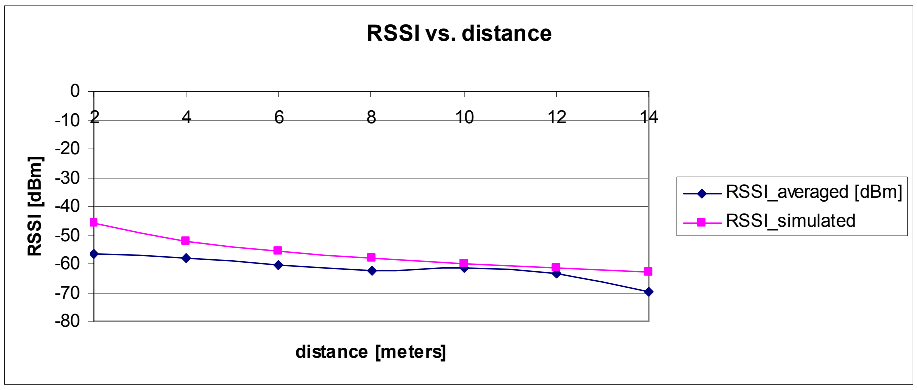


Figure 5.24: Arduino: RSSI vs. distance, set I

The error between the averaged RSSI and the simulated RSSI varies between 20 dBm and 0 dBm, this being in a range similar to the momentarily RSSI values presented before.

Fig. 5.24 presents the averaged received power. As for the momentarily values, it can be observed that the RSSI slope and the received power slope for averaged values are similar.

Table 5.17 presents a second set of averaged RSSI values. As for the previous averaged set, the variation of RSSI with distance is presented in Fig. 5.25. The values of the averaged RSSI decrease with distance, which is expected from theory.

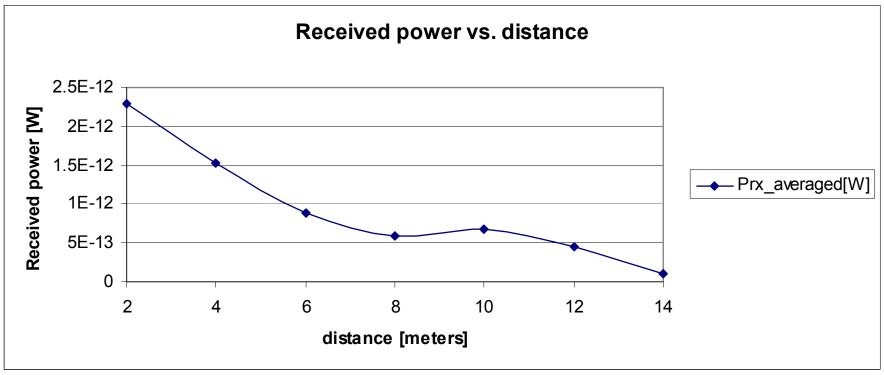


Figure 5.25: Arduino: Averaged received power vs. distance, set I

The error between the averaged and the simulated RSSI varies between -15 and 0 dBm. This is slightly reduced when compared with the previous set. However, when comparing this error with the error obtained for the momentarily RSSI it can be noticed that the error range is similar.

Table 5.17 Arduino average RSSI, set II

Distance [meters]	**RSSI$_{averaged}$ [dBm]**	**P$_{Rx}$[Watt]**
2	-56.97	$2.00\ 10^{-9}$
4	-56.97	$2.00\ 10^{-9}$
6	-62.24	$5.96\ 10^{-10}$
8	-62.24	$5.96\ 10^{-10}$
10	-62.82	$5.21\ 10^{-10}$
12	-62.82	$5.21\ 10^{-10}$
14	-70.43	$9.03\ 10^{-11}$

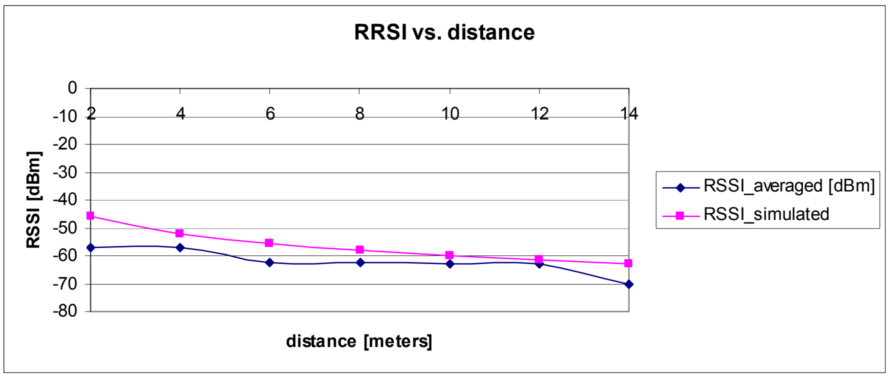


Figure 5.26: Arduino: averaged RSSI vs. distance, set II

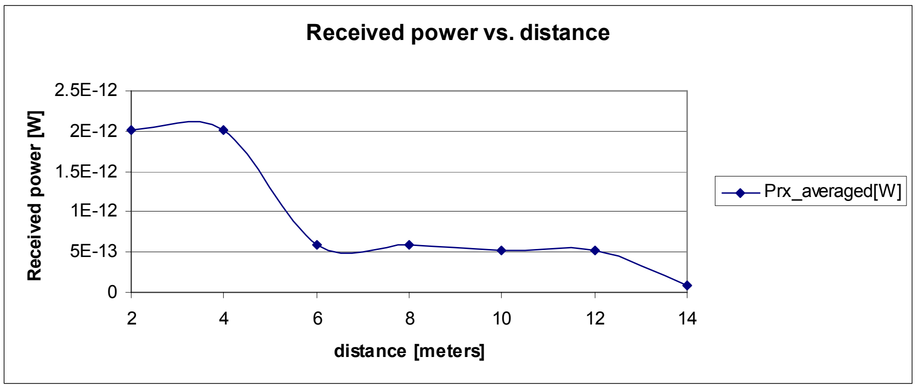


Figure 5.27: Arduino: Averaged received power vs. distance, set II

As before, the range of the averaged received power is maintained in the limits of the momentarily received power. In addition to this, the averaged values are smaller than the simulated values. The averaged received power decreases smoothly as the distance increases and the slopes of the simulated and averaged values are similar. This stands for the averaged and the momentarily values as well.

At a first look there is no significant improvement for the RSSi values when using the average method. The ranges for RSSI and received power are similar, the errors between the measured and the simulated values are similar. By using the averaged values in a real life localization scenario the differences could be more obvious. The fact that the difference between the averaged and the momentarily values is small proves that the Arduino PWM measurement algorithm is accurate, as well as the fact that RSSI momentarily measurements are almost as accurate as averaged measurements for open space scenarios.

The next section presents localization results based averaged RSSI. The results are presented for the same algorithms as for momentarily RSSI values.

The network is composed of 4 anchors and one sensor and it is placed in free space. For the communication between nodes the Arduino Duemilenove and Xbee nodes are used. The communication is in broadcast mode. Using the values of RSSI the distance between the network nodes is determined. The anchor positions are known. Using all this information the algorithms simulated in chapter 3 are applied and the position of the sensor is determined. The surface on which the network spreads is a 10 x10 meters, with the sensor being placed in the center of this surface. In what follows the results obtained for each board are presented from two points of view: the localization error and the sensor positions.

The sensor in placed in the center of the network, which corresponds to the $S(0,0)$, while the anchors are placed as follows: $A_1\,(0,10)$, $A_2\,(10,0)$, $A_3\,(0,-10)$ and $A_4\,(-10,0)$.

5.6.1. Maximum Likelihood Estimation

The results for MLE localizarion using averaged RSSI are presented in Table 5.18.

Table 5.18 Averaged RSSI: MLE positioning

	x	y
$S_{measured}$	0.84	0.44
$d(S_{real}, S_{measured})$ [meters]	0.95	

A comparative analysis of the averaged RSSI positioning error and the momentarily RSSI positioning errors is presented in Table 5.19.

Table 5.19 MLE positioning error analysis

Averaged RSSI positioning error [meters]	**Momentarily RSSI positioning error [meters] Set I**	**Momentarily RSSI positioning error [meters] Set II**
0.95	1.27	0.02

The analysis of Table 5.19 shows that the averaged RSSI error is situated between the errors for momentarily set I and II. Such a situation is determined by the succession of RSSI readings. Therefore, averaging can sometimes be translated into a reduction of the positioning error or the opposite.

The positioning error obtained by averaging consecutive RSSI values is small enough to be used in applications which require a high level of accuracy, but not small enough to be used in critical safety applications.

5.6.2. Weighted Centroid Localization

WCL localization results using averaged RSSI are presented in Table 5.20.

Table 5.20 Averaged RSSI: WCL positioning

	x	y
$S_{measured}$	0.91	1.48
$d(S_{real}, S_{measured})$ [meters]	1.75	

In order to get a better image on the effect of averaging on the obtained results the comparative analysis in Table 5.21 is necessary.

By analyzig the results a situation similar to the MLE is observed. The averaged RSSI positioning error is situated in between the momentarily values. The cause for this stands in the RSSI values, which, as presented in the early stages of this chapter vary: sometimes incresing other times decresing.

Table 5.21 WCL positioning error analysis

Averaged RSSI positioning error [meters]	**Momentarily RSSI positioning error [meters] Set I**	**Momentarily RSSI positioning error [meters] Set II**
1.75	3.90	0.67

5.6.3. Malguki Spring

The positioning error for Malguki spring localization using averaged RSSI is presented in Table 5.22.

Table 5.22 Averaged RSSI: Malguki spring positioning

	x	**y**
$S_{measured}$	0.48	0.22
$d(S_{real}, S_{measured})$ [meters]	0.53	

The positioning error obtained in this case is small and offers a good perspective of using the algorithm in applications that require an increaesd level of safety. However, as for the previous algorithms a comparative anlysis with the momentarily results is necessary. The analysis is found in Table 5.23.

Table 5.23 Malguki spring positioning error analysis

Averaged RSSI positioning error [meters]	**Momentarily RSSI positioning error [meters] Set I**	**Momentarily RSSI positioning error [meters] Set II**
0.53	2.01	0.46

The positiong error for averaged RSSI is situated between the momentarily errors. This is due to the variations of the RSSI readings.

5.6.4. Modified Multidimensional Scaling

Table 5.24 presents the positioning results for MMDS algorithm:

Table 5.24 Averaged RSSI: MMDS spring positioning

	x	**y**
$S_{measured}$	0.01	0.01
$d(S_{real}, S_{measured})$ [meters]	0.01	

Even though the results in Table 5.24 are very accurate, it is necessary to do a comparative anlysis with the positioning errors obtained using momentarily values. This analysis can be found in Table 5.25.

Table 5.25 MMDS positioning error analysis

Averaged RSSI positioning error [meters]	**Momentarily RSSI positioning error [meters] Set I**	**Momentarily RSSI positioning error [meters] Set II**
0.01	0.01	0.01

The analysis in Table 5.25 shows that the positioning error for both the averaged RSSI and the momentarily RSSI is extremely reduced. This can be determined either by the high accuracy of this algorithm or by the successive readings.

5.6.5. Second Order Cone Programming

This sub-section presents SOCP positioning results for averaged RSSI. The results are presented in Table 5.26.

The positioning error obtained by using averaged RSSI is very small: just 0.03 meters. Such a value recommends this algorithm for use in applications with an increased degree of safety. However, a clearer image on the algorithm performances using averaged and momentarily RSSI can be obtained by means of Table 5.27.

Table 5.26 Averaged RSSI: SOCP positioning

	x	**y**
$S_{measured}$	0	0.03
$d(S_{real}, S_{measured})$ [meters]	0.03	

Table 5.27 SOCP positioning error analysis

Averaged RSSI positioning error [meters]	**Momentarily RSSI positioning error [meters] Set I**	**Momentarily RSSI positioning error [meters] Set II**
0.03	0.01	0.01

The momentarily RSSI positioning errors are very small indeed, comparative to the averaged RSSI positioning error. This confirms the algorithm accuracy: the positioning error is similar for both momentarily and averaged RSSI values.

5.6.6. Semidefinite Programming

SDP positioning results are presented in Table 5.28

Table 5.28 Averaged RSSI: SDP positioning

	x	**y**
$S_{measured}$	-2.55	0.24
$d(S_{real}, S_{measured})$ [meters]	2.57	

The positioning error obtained in this case is high compared to the previous results. In order to complete the anlysis, the data in Table 5.29 must be analyzed.

The momentarily RSSI positioning errors are smaller than the averaged RSSI positioning error. A cause for this can represent the successive RSSI readings, which can present higher or lower variations.

Table 5.29 SDP positioning error analysis

Averaged RSSI positioning error [meters]	**Momentarily RSSI positioning error [meters] Set I**	**Momentarily RSSI positioning error [meters] Set II**
2.57	0.67	0.39

5.6.7. Averaged RSSI Conslusions

Section 5.5 presents the problem of localization from the point of view of averaging RSSI readings. The author presents the approach from two points of view: the distance between a transmitter and a receiver and from the point of view of localization algorithms.

Sections 5.5.1- 5.5.6 presented thelocalization results using averaged RSSI values. The averging is based on 10 consecutive RSSI readings. All the localization algorithms presented before were analyzed form this new perspective.

A particular case of positioning errors is represented by the case when the compared values are identical or very similar (SOCP, Malguki spirng). This can be due, as before, to the RSSI readings or to the good accuracy of the considered algorithm.

5.7. Other Localization Scenarios

This section presents localization results for other localization scenarios. These scenarios are derived from the one introduced in section 5.5. The anchors have the same positions as before, but the sensor position is varied. The results presented here are for averaged RSSI over 100 samples.

5.7.1. Scenario I

The sensor in placed at $S(10,10)$, while the anchors are placed as follows: $A_1(0,10)$, $A_2(10,0)$, $A_3(0,-10)$ and $A_4(-10,0)$. The algorithms applied to the network are: maximum likelihood estimation, weighted centroid localization, Malguki spring, modified multidimensional scaling, second order cone programming and semidefinite programming.

The results for all the considered algorithms are presented in Table 5.30

The errors obtained for this new configuration are larger than the error obtained for the previous set-up. The algorithm that provides the smallest positioning error for this scenario is WCL, while Malguki and MMDS provide the highest positioning errors.

Table 5.30 Scenario I positioning results

Algorithm	$S_{measured}$		$d(S_{real}, S_{measured})$ [meters]
	x	**y**	
MMDS	3.32	5.31	8.16
Malguki spring	5.53	6.23	5.65
SOCP	7.42	10.87	2.72
SDP	9.12	8.39	1.83
MLE	8.91	10.40	1.16
WCL	10.77	10.51	0.92

5.7.2. Scenario II

The sensor in placed at $S(6,2)$, while the anchors are placed as follows: $A_1(0,10)$, $A_2(10,0)$, $A_3(0,-10)$ and $A_4(-10,0)$. The algorithms applied to the network are: maximum likelihood estimation, weighted centroid localization, Malguki spring, modified multidimensional scaling, second order cone programming and semidefinite programming. The positioning results for these algorithms are presented in Table 5.31.

Compared to the scenario introduced in section 5.7.1, the positioning error does not cover a very wide range of values. The smallest positioning error is obtained for SOCP and it is 2.14 meters. The largest positioning error is 5.18 meters and it is obtained for Malguki spring.

Table 5.31 Scenario II positioning results

Algorithm	$S_{measured}$		$d(S_{real}, S_{measured})$
	x	**y**	[meters]
Malguki spring	6.85	7.11	5.18
WCL	9.99	0	4.46
SDP	5.07	-1.61	3.72
MLE	2.84	1.96	3.16
MMDS	3.08	1.84	2.92
SOCP	4.17	3.11	2.14

5.7.3. Scenario III

The sensor in placed at $S(0,-6)$, while the anchors are placed as follows: $A_1(0,10)$, $A_2(10,0)$, $A_3(0,-10)$ and $A_4(-10,0)$.
The algorithms applied to the network are: maximum likelihood estimation, weighted centroid localization, Malguki spring, modified multidimensional scaling, second order cone programming and semidefinite programming. The positioning results for these algorithms are presented in Table 5.32.

As for sections 5.7.1 and 5.7.2, the positioning errors in Table 5.32 are larger that the posirioning errors obtained in section 5.5. Tha current positioning error does not have a large range of variation. The smallest positioning error is obtained for SOCP (3.63 meters), while the highest positioning error is obtained for SDP (8.62 meters). The variation range of the positioning error for the current scenario is similar to the range in section 5.7.2.

Table 5.32 Scenario III positioning results

Algorithm	$S_{measured}$		$d(S_{real}, S_{measured})$
	x	**y**	[meters]
SDP	2.11	2.36	8.62
Malguki spring	1.20	-7.94	5.20
MLE	0.82	-1.47	4.60
MMDS	0.38	-2.06	3.95
WCL	0.02	-9.99	3.90
SOCP	2.12	-3.05	3.63

5.7.4. Other Localization Scenarios: Conclusions

Section 5.7 presented three localization scenarios that did not have the sensor node placed in the center of the network. The anchors are placed identically to sections 5.5 and 5.6, but the sensor node is moved within the area of the 10 x 10 meter room.

The positioning results are presented in tables, in decreasing order. The values for the positioning errors are higher than in the case when the sensor node is placed in the center of the network. Such a situation was expacted, since moving the sensor node further from the center of the network resulted in an increase in the positioning errors. However, the values obtained for the positioning errors are similar to the values presented by other authors and which can be consulted in Chapter 3, section 3.6.

Considering these network scenarios and the one used in sections 5.5 and 5.6, it can be concluded that the modern localization algorithms proved to be effective. At the same time, modern localization algorithms represent a tradeoff between precision, computational time and optimization.

5.8. Conclusions

In this chapter the author built an experimental set-up composed of a transmitter and a receiver, as well as a small scale network on which the studied algorithms were verified. A simple transmitter-receiver network was built. The nodes were made up of Xbee transceivers and APIC development boards. Since the results were unreliable, it was necessary to use another measurement method: the Arduino Duemilenove with Xbee shield. The Xbee shield is an extension of the Arduino board dedicated for Xbee module drive. The advantage of this board over the APIC is its built-in PWM measurement function, which proved to be more accurate and reliable than the one using APIC.

The scope of the measurement method was to verify if the duty cycle, RSSI and received power respect the theoretical features. Theory stated that DC, RSSI and received power should decrease once the distance between the transmitter and the receiver increases. In order to determine the variation of the mentioned parameters the distance between the transmitter and the receiver was varied to meters at a time until a maximum range of 14 meters. For each step the author recorded the duty cycle. Based on it the RSSI and the received power were calculated.The next step in the analysis was the graphical representation of the measured RSSI vs. the theoretical RSSI. The graphs confirmed the theory: the duty cycle, RSSI and received power decreased with distance. The error between the measured and the simulated RSSI is around 10-15 dBm maximum, but there were situations where there is no significant error between theory and practice.

Since the rest of the results were similar with those presented in this chapter and the results respected the theoretical aspects of the problem the author considered that a selection of all the result would serve the purpose

The small scale network used for the experiments is built of 5 Xbee nodes: 4 anchors (nodes with known positions) and 1 sensor (with unknown position). The Xbee transceivers were driven using Arduino boards. Based on the RSSI values the distances between anchors and sensors were determined. Having as starting point the distances between anchors and sensors and the anchor positions four different positioning algorithms were applied by the author. The result of these algorithms is a set of coordinates representing the sensor position and a positioning error. The network topology is presented for each situation, so that a picture of the results exists.

Another point of view from which it would be interesting to analyze the algorithms is the real distances between anchors and sensors and the measured

(RSSI based) anchor-sensor distances. This analysis is conducted by the author using Table 5.33 (for set one) and Table 5.34 for set 2.

For both sets of data the error between the real and the measured anchor-sensor distances are well under 1 meter. Still, exceptions exist, when the error is 0.99 meters. Keeping these aspects in mind it can be concluded that the measured anchor- sensor distances are in acceptable limits and for this reason the results obtained after applying all the algorithms are in full accordance.

Table 5.33 Anchor-sensor distances, set I

Real distance [meters]	**Measured distance [meters]**	**Error [meters]**
10	10.81	0.81
10	12.74	2.74
10	10.99	0.99
10	10.17	0.17

Table 5.34 Anchor-sensor distances, set II

Real distance [meters]	**Measured distance [meters]**	**Error [meters]**
10	10.99	0.99
10	10.17	0.17
10	10.99	0.99
10	10.42	0.42

As pointed out throughout this chapter the algorithm complexity determines the computational time. This is an aspect to consider in any application: the faster the algorithm the faster the feedback. The computational time analysis is performed by the author in Table 5.35.

The computational time analysis reveals that SOCP is the slowest out of all 6 algorithms studied by the author. On second place is WCL and on third place SDP. The common point for these algorithms is the use of optimization, in a form or another, which is complex and can require many iterations before converging, making it a time consuming calculus method. On the opposite side, the fastest algorithm proves to be Malguki spring. In the middle positions MLE and MMDS are found. These algorithms use direct calculus to obtain the results and therefore small computational times are expected.

After applying all the four algorithms to the considered topologies and calculating the positioning error the results can be synthesized in Table 5.32. The results are a selection of all the presented results.

Considering the positioning errors in Table 5.26, a classification of the algorithms can be made. The smallest positioning errors are provided by SOCP. The next algorithms from the positioning error point of view are MMDS and SDP. On the fifth position is SDP. On the sixt position is Malguki spring; this algorithm offers acceptable errors, though high, while the highest errors are provided by Malguki spring. The MLE uses direct calculus and the results obtained after applying it are predictable. The same is valid for Malguki and WCL which are optimization-based algorithms. Even though MMDS uses direct calculus and it is a slow algorithm due to the complex calculations it performs the obtained results are very good. The observations made for the first set of data are valid for the second set of measurements.

Table 5.35 Algorithm computational time

Algorithm	Computational Time [seconds]
SOCP	2.60
WCL	1.93
SDP	1.68
MLE	1.63
MMDS	0.88
Malguki	0.73

5.36 Algorithm ranking

Algorithm	$d(S_{real}, S_{measured})$ [meters]
SOCP	0.01
MMDS	0.01
SDP	0.67
MLE	1.27
Malguki	2.01
WCL	3.90

A careful analysis of Table 5.35 (computational time) and 5.36 (positioning errors) it can be seen that the algorithms providing the best results are not necessarily the faster. The values of the computational times increase proportionally to the network size: more anchors and sensors means more time for processing.

In order to obtain a tradeoff between speed and computational time the author introduces an **original evaluation criterion** for the algorithms. This criterion is based on the computational time and the error provided by the algorithm. The criterion is:

$$c = error \cdot \frac{1}{computational\_time} \tag{5}$$

By applying (5) to the results from Tables 5.35 and 5.36 the algorithm ranking in Table 5.37 is obtained. Table 6.17 reveals a certain change in the algorithm hierarchy, especially for the last positions. On the first two places the algorithms that provided the smallest errors are to be found: SOCP and MMDS. On the Third place, SDP maintains its position form Tables 5.35 and 5.36. However, WCL is becomes the last but one algorithm from Table 5.37. Compared to Table 5.35 it is a downgrade, but compared to Table 5.36 it is progress, which proves that the original criterion is functional. The observation can be maintained for Malguki spring algorithm as well. All in all, at a close look, the hierarchy in Tables 5.35 and 5.36 has changed with the introduction of the original criteria.

After having applied the algorithms on a small scale real life network it can be concluded that the choice for one of these algorithms can be made based on certain criteria. The first criterion to be kept in mind is the application where the algorithm will be used: whether it requires high precision or not. The second aspect to be considered is the computational time. This aspect is tied to the application: does the application need to respond fast to stimuli or not? A third aspect is concerned with hardware dimensions: the computational complexity of the algorithm determines the resources necessary for the calculations to be performed in the smallest time possible. All in all, when making a choice for an algorithm or the other a tradeoff must be made between all the aspects presented earlier.

Table 5.37 Algorithm classification with "c" criterion

Algorithm	**c**
SOCP	0.03
MMDS	0.11
SDP	0.39
MLE	0.77
WCL	2.02
Malguki	2.75

Sections 5.5.1- 5.5.6 presented the localization results using averaged RSSI values. The averging is based on 10 consecutive RSSI readings. All the localization algorithms presented before were analyzed form this new perspective.

The comparative analysis of the averaged RSSI positioning results and the momentarily RSSI positiong error revealed that in some cases (MLE, WCL) the positioning error was placed between the momentarily errors, while in other cases it was higher (SDP). The cause for this can be the successive RSSI readings, which increase and decrease without a known law of variation.

After a thorough analysis of all the algorithms a classification was made by the author. The most feasible algorithm proved to be MMDS. Despite its complexity and time and resources consuming, this algorithm provides the smallest localization errors. On second place Malguki spring algorithm can be found, while on third place MLE is positioned. The algorithm that is least recommended in real life applications is WCL.

A particular case of positioning errors is represented by the case when the compared values are identical or very similar (SOCP, Malguki spirng). This can be due, as before, to the RSSI readings or to the good accuracy of the considered algorithm.

5.9. Contributions

The author's main contributions to this chapter are mainly practical contributions.

The first contribution is conceiving and building the experimental set-up (building of the network nodes) and conceiving the measurement method.

Another contribution is the Matlab code necessary for calculating RSSI and received power based on the duty cycle.

A third contribution is the result analysis: graphs and conclusions.

The implementation of the APIC C code for driving the Xbee transceivers is another contribution.

Other contributions include:

- The processing of the duty cycle values in order to obtain the RSSI. The code for processing the data was written by the author in Matlab.
- The Matlab implementation of all of the four localization algorithms
- Development of a GUI that groups all the algorithms and displays the results in a grouped manner
- The analysis of the positioning results

- The ranking of the algorithms given the positioning errors
- The ranking of the algorithms considering computational time
- Analysis of the aspects to be kept in mind when making a choice for one of the algorithms

The author also proposes in this chapter an analysis of localization based on averaging RSSI values. This analysis is performed by the author from the point of view of the distance between transmitter and receiver, and from the point of view the considered positioning algoriothms. When discussiong localization results the author proposes a comparative analysis between the averaged positioning results and the momentarily positioning results.

5.10. References

1. [Des07] Jasmin Desai and Uf Turelli, "Evaluation Performance of Various Localization Algorithms in Wireless and Sensor Networks", 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio communications, 2007
2. [Che05] K.W. Chenung and H.C. So, "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements", IEEE Transactions on Signal Processing, Vol. 53, No.2, pp. 460-470, February 2005
3. [Ari04] Jagoba Arias et. al., "GPS-less location algorithm for wireless sensor networks", Elsevier Computer Communications, vol. 30, pp. 403-409, 2004
4. [Fin11] Andreas Fink and Helmut Beikirch., "Reliable Radio-based Human Tracking in Hazardous Environments", Wireless Congress- 2011 systems and applications, Munich, Germany, November 2011
5. [Jan11] Adel Javanmard and Andrea Montari, "Localization from Incomplete Noisy Distance Measurements", 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31 July-5 August 2011, pp. 1584-1588
6. [Rus10_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Maximum Likelihood Estimation Algorithm Evaluation for Wireless Sensor Networks", SYNASC 2010, Sept. 2010, pp. 95-98
7. [Rus10_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Target detection algorithm validation in WSN", ISETC 2010, Nov. 2010, pp. 373-376
8. [Rus11_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Evaluation of wireless sensor networks localization algorithms", IDAACS 2011, Sept. 2011, pp. 857-862
9. [Rus11_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Evaluation of some node localization algorithms", SIITME 2011, Oct. 2011, pp. 287-290
10. [Rus11_03] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Performance analysis of wireless sensor network node localization algorithm", SCVT 2011, Nov. 2011, pp. 1-5
11. [Rus12_01] **Rusnac, Ruxandra-Ioana** and Jivulescu, Maria Anastasia, Wireless Sensor Network Node Localization via Second Order Cone Programming, in review: ISETC 2012
12. [Van11] Vanheel et al. "Automated linear regression tools improve RSSI WSN localization in multipath indoor environment", EURASIP Journal on Wireless Communications and Networking 2011 2011:38.
13. [Rus12_02] **Rusnac, Ruxandra-Ioana** and Gontean Aurel, "Wireless Sensor Network Real Life Localization", under review IJCT

14. [Nad08] Nadimidi, E.S et. al, “ZigBee-based wireless sensor networks for monitoring animal presence and pasture time in a strip of new grass”, Elsevier Computer and Electronics in Agriculture, (61) 2008, pp. 79-87
15. [Tso09] Yu-Tso et. al, "A RSSI-based Algorithm for Indoor Localization Using ZigBee in Wireless Sensor Network”, Proceedings of the 15th International Conference on Distributed Multimedia Systems (DMS 2009) , San Francisco, USA, Sep. 2009, pp. 70-75.
16. [Fin09] Fink, Andres, Beikirch, H "RSSI-based indoor localization using antenna diversity and plausibility filter”, 2009 6th Workshop on Positioning Navigation and Communication (2009), pp. 159-165
17. [Hyo09] Ahn, H-S et. al., "Environmental-Adaptive RSSI-Based Indoor Localization”, IEEE Transactions on Automation Science and Engineering, Volume: 6, Issue: 4, 2009, pp. 1-8
18. [Gol10] Goldoni,Emanuele et. al, "Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4”, 2010 European Wireless Conference EW (2010), pp. 71-77
19. [Xbe11] XBee®/XBee-PRO® ZB RF Modules, Datasheet and user manual, Digi International, 2011
20. [Ard11] Arduino Datasheet and User Manual: http://arduino.cc/en/Main/arduinoBoardDuemilanove
21. [Van11] Vanheel et al. “Automated linear regression tools improve RSSI WSN localization in multipath indoor environment”, EURASIP Journal on Wireless Communications and Networking 2011 2011:38.
22. [Zur09] Richard Zurawski, “Networked Embedded Systems”, CRC Press, chapter 3, pp. 3-6, 2009
23. [Hol05] Holger Karl and Andreas Willig, “Protocols and Architectures for Wireless Sensor Networks” Wiley Interscience, pp. 17-67, 2005
24. [Rap02] Theodore S. Rappaport, “Wireless Communications- Principles and Practice”, Prentice Hall Communication Engineering and Technologies Series, pp. 81-138, 2002
25. [Ers05] ETSI EN 300 328 V1.7.1, 2006

6. CONCLUSIONS AND CONTRIBUTIONS

In the thesis the author focused on the problem of wireless sensor network node localization. The reason for which the author chose this topic resides in the expansion of wireless sensor networks in every day life, but especially in the future that lies ahead for these networks. The author believes that in a few years' time wireless sensor networks will be implemented in automotive industry, especially in smart cars and in intelligent traffic control. Smart cars are the cars of the future, which will be able to communicate with each other, that could adjust their route and speed depending on the traffic conditions communicated by other cars on the road.

The problem of localization was studied by the author from three points of view. The first point of view was the theoretical aspect. It was necessary to identify how research in this area stands, what has been done and what needs to be done. The second aspect that was considered was the simulation aspect: before proceeding to a practical study it was necessary to see what localization algorithms exist and how promising they are: whether it was worth studying them or not. For this original Maltab simulations were done by the author and the algorithms considered to be best suited for the purpose were studied. The third aspect was the practical aspect: a small scale network was built and the author gathered the necessary information for implementing the simulated algorithms. The localization algorithms were ranked having in mind different aspects that could influence the choice for one algorithm or the other.

All the three aspects mentioned above were detailed in five chapters.

Chapter 1 was thought as an introduction to wireless sensor networks in general and to localization in particular. In as far as the general introduction to wireless sensor networks is concerned general aspects as applications of the networks as well as the structure of a network node were presented. The author presented a careful analysis of the modeling possibilities. The modeling can be done at network level and at node level. Different approaches encountered in literature were presented by the author through a comprehensive literature survey.

A survey of wireless sensor networks for location detection was done. At this point the focus was on the mobile agent paradigm and the importance it has for this topic. The problem of localization could not be complete without a survey of the most important node localization algorithms. The survey was done via a classification of these algorithms in two branches, classical algorithms and modern algorithms.

Throughout this chapter it was proven that wireless sensor networks are very complex and their analysis comprises many aspects. A special focus was made on the problem of localization, which is the theme of the thesis.

In **Chapter 2** the author presented an analytical survey of some of the most popular, more complex and more challenging localization algorithms encountered in literature. The algorithms considered for analysis were the Maximum likelihood estimation, the weighted centroid localization, the iterative trilateration, Malguki spring and modified multidimensional scaling. For each algorithm the steps necessary for implementation were presented.

In section 2.8 a comparative analysis of the algorithms was performed by the author. This analysis was introduced by means of a table and it considered such aspects as the algorithm complexity, if the algorithm uses optimization or not and the algorithm computational time. Such an analysis is very important before making a choice for an algorithm as it involves safety aspects, node design aspects etc. Another aspect to consider is the network size, which influences the speed and type of the algorithm to use.

In section 2.7 simulation scenarios used for the algorithm analysis were introduced. Together with this the pathloss model simulation result is presented. The slope of the pathloss model is as expected from literature and the pathloss decreases as distance increases.

In **Chapter 3** the author presented original simulation results. The code for all the algorithms surveyed in Chapter 2 was written by the author in Matlab. The simulation scenarios studied by the author were the random and the uniform configurations. Since the results (positioning error) provided by the algorithm required some improvements the author came up with a new simulation scenario. This scenario was named hybrid and to the author's knowledge such a configuration was not encountered anywhere in literature. The idea for this novel scenario came at a careful analysis of the error obtained using random and uniform anchors. The hybrid configuration combined the two classical scenarios in the sense that four anchors are placed at an equal distance from the corners of the space, while the rest are deployed randomly in the remaining surface. Such an arrangement served the author's purpose, bringing, for most of the studied algorithms the desired error reduction.

The algorithms studied in this chapter have different complexities. The simplest ones use direct calculus while the most complicated ones make use of optimization methods. This has impact on the implementation complexity and it represented a challenge. As pointed out throughout the chapter the author obtained better positioning errors than other authors have obtained in similar conditions.

Chapter 4 was focused on a mathematical approach for the problem of localization: second order cone programming and semidefinite programming localization of wireless sensor network nodes nodes. The problem was given a thorough theoretical analysis, as well as a Matlab implementation. The results were analyzed by the author using three different anchor placement strategies: the classical random placement, the uniform placement, as well as novel scenario called hybrid placement. The careful analysis revealed both advantages and disadvantages of the considered scenarios, as well as the influence that the number of anchors has over the results.

The work aimed to determine an error reduction mechanism for the random and uniform placements. The goal was achieved through the hybrid anchor placement method and thus the errors obtained using this approach were smaller.
When comparing the results obtained for the current algorithms with the weighted centroid localization, the iterative trilateration and the Malguki spring algorithms the errors are similar when using both small and high numbers of anchors, no matter which anchor placement scenario is used. However, when comparing the results with other authors, the current work provides better results. The algorithms with which the SOCP approach was compared are classical algorithms. This proves that the choice for classical or modern method remains the users' choice, as there are no major differences in the results obtained with either method.

In **Chapter 5** the author built an experimental set-up that would be the starting point for large scale networks. A simple transmitter-receiver network was built. The nodes were made up of Xbee transceivers and APIC development boards. Since the results were unreliable due to the repetitive values of the duty cycle at different distances. Therefore it was necessary to use another measurement method: the Arduino Duemilenove with Xbee shield. The Xbee shield is an extension of the Arduino board dedicated for Xbee module drive. The advantage of this board over the APIC is its built-in PWM measurement function.

The scope of the measurement method was to verify if the duty cycle, RSSI and received power respect the theoretical features. Theory stated that DC, RSSI and received power should decrease once the distance between the transmitter and the receiver increases. In order to determine the variation of the mentioned parameters the distance between the transmitter and the receiver was varied to meters at a time until a maximum range of 14 meters. For each step the author recorded the duty cycle. Based on it the RSSI and the received power were calculated.

The next step in the analysis was the graphical representation of the measured parameters vs. the theoretical values. The graphs confirmed the theory: the duty cycle, RSSI and received power decreased with distance. What is even more is that the error between the measured and the simulated RSSI is around 10-15 dBm maximum, but there are situations where there is no error between theory and practice.

In this chapter the author presented a selection of the obtained results. Since the rest of the results were similar with those presented in this chapter and the results respected the theoretical aspects of the problem the author considered that a selection of all the result would serve the purpose.

In this chapter a real life localization scenario is introduced. For this, the author built a small scale real life localization scenario. The network used for the experiments is built of 5 Xbee nodes: 4 anchors (nodes with known positions) and 1 sensor (with unknown position). The Xbee transceivers were driven using Arduino boards. Based on the RSSI values the distances between anchors and sensors were determined. Having as starting point the distances between anchors and sensors and the anchor positions four different positioning algorithms were applied by the author. The result of these algorithms is a set of coordinates representing the sensor position and a positioning error. The network topology is presented for each situation, so that a picture of the results exists.

After a thorough analysis of all the algorithms a classification was made by the author. The smallest positioning errors are provided by SOCP. The next algorithms from the positioning error point of view are MMDS and SDP. On the fifth position is SDP. On the sixt position is Malguki spring; this algorithm offers acceptable errors, though high, while the highest errors are provided by Malguki spring. The MLE uses direct calculus and the results obtained after applying it are predictable. The same is valid for Malguki and WCL which are optimization-based algorithms. Even though MMDS uses direct calculus and it is a slow algorithm due to the complex calculations it performs the obtained results are very good. The observations made for the first set of data are valid for the second set of measurements.

After having applied the algorithms on a small scale real life network it can be concluded that the choice for one of these algorithms can be made based on certain criteria. The first criterion to be kept in mind is the application where the algorithm will be used: whether it requires high precision or not. The second aspect to be considered is the computational time. This aspect is tied to the application: does the application need to respond fast to stimuli or not? A third aspect is concerned with hardware dimensions: the computational complexity of the algorithm determines the resources necessary for the calculations to be performed in the smallest time possible. This can have an impact on the size of the sensor node, if considering to build the node in its own enclosure, not as a prototype as the author used for the experiments. All in all, when making a choice for an algorithm or the other a tradeoff must be made between all the aspects presented earlier.

Throughout the thesis the author had contributions to each of the five chapters. In what follows the **contributions** to each chapter are presented.

The author's contributions for **Chapter 1** are mainly theoretical contributions. The chapter was meant as in introduction to wireless sensor networks in general and to localization in particular. In as far as the general introduction to wireless sensor networks is concerned general aspects as applications of the networks as well as the structure of a network node were presented. A careful analysis of the modeling possibilities was performed. The modeling can be done at network level and at node level. Different approaches encountered in literature were analyzed.

A survey of wireless sensor networks for location detection was done. At this point the focus was on the mobile agent paradigm and the importance it has for this topic. The problem of localization could not be complete without a review of the most important node localization algorithms. This was accomplished via a classification of these algorithms in two branches, classical algorithms and modern algorithms.

Throughout this chapter it was shown that wireless sensor networks are very complex and their analysis comprises many aspects. A special focus was made on the problem of localization, which is the theme of the thesis.

The author's contributions to **Chapter 2** are theoretical contributions.

A contribution is the analysis of the localization algorithms. The algorithms are presented from a theoretical point of view and the most important aspects of each of the considered algorithms are highlighted.

Another contribution is the original comparative analysis of the considered algorithms. This analysis is made by means of Table 2.2. Here three aspects are considered: the algorithm complexity, if the algorithm uses optimization pr not and the computational time. When performing the analysis the author highlights the tradeoff that must be made between computational time and algorithm complexity when thinking of real life applications.

Obtaining the computational times of each algorithm is done by the author by means of original Matlab simulations.

Since **Chapter 3** is focused on the implementation of the algorithms introduced in Chapter 2, the author's main contributions to this chapter are practical (applicative) contributions.

One contribution is the Matlab original simulations: the code written to implement the algorithms as well as the implementation of the pathloss model. From this code the second contribution is obtained: the original simulation scenario: the original hybrid set-up, which, to the author's knowledge was not encountered anywhere in literature

Another contribution is the comparative anlysis of the author's results and other author's results. The analysis is performed in section 3.6 by means of tables, for several of the studied algorithms. The comparative anlysis proves that the results obtained by the author are better than other results presented in literature: the positiong errors are similar or smaller.

Chapter 4 contained a modern approach to the problem of node localization: the second order cone programming and semidefinite programming. The main contributions to this chapter are both theoretical and applicative.

From the point of view of the SDP programming approach a theoretical contribution is the reformulating of the problem of localization for second order cone programming. This is done by introducing the upper limit δ and rearranging the newly obtained equations in such a way that the problem can be implemented into Matlab via SeDuMi toolbox. The contributions to SDP also include the original Matlab simulations and the analysis of the results, which can be included in the applicative contributions to this chapter.

The contributions regarding SOCP include the obtaining of the analytical solution to the problem of semidefinite programming localization, which is a theoretical contribution, and the original Matlab simulations. The analysis of the obtained results is another contribution to this chapter. The original simulations and the result analysis is an applicative contribution to this chapter. The development of the hybrid anchor placement and its verification by means of simulation is one of the author's contributions to this chapter. The original hybrid placement is both a theoretical and an applicative contribution to the current chapter.

Chapter 5 is introduces the experimental set-up as well as presents a real life localization scenario. Therefore the author's main contributions to this chapter are mainly practical contributions.

The first contribution is conceiving and building the experimental set-up (building of the network nodes) and conceiving the measurement method.

Another contribution is the Matlab code necessary for calculating RSSI and received power based on the duty cycle.

A third contribution is the result analysis: graphs and conclusions.

The implementation of the APIC C code for driving the Xbee transceivers is another contribution.

Other contributions include:

- The processing of the duty cycle values in order to obtain the RSSI. The code for processing the data was written by the author in Matlab.
- The Matlab implementation of all of the four localization algorithms
- Development of a GUI that groups all the algorithms and displays the results in a grouped manner
- The analysis of the positioning results
- The ranking of the algorithms given the positioning errors
- The ranking of the algorithms considering computational time
- Analysis of the aspects to be kept in mind when making a choice for one of the algorithms
- the introduction of the "c" classification criterion, which acts as a mediator between computational time and positioning error.

The author also proposes in this chapter an analysis of localization based on averaging RSSI values. This analysis is performed by the author from the point of view of the distance between transmitter and receiver, and from the point of view the considered positioning algoriothms. When discussiong localization results the author proposes a comparative analysis between the averaged positioning results and the momentarily positioning results.

In addition to the contributions listed above the author published a number of 6 published papers:

1. [Rus10_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Maximum Likelihood Estimation Algorithm Evaluation for Wireless Sensor

Networks", Proceedings of 2010 12th International Symposium on Numeric, Scientific and Symbolic Computing (SYNASC 2010), Sept. 2010, Timişoara, Romania, pp. 95-98
2. [Rus10_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Target detection algorithm validation in WSN", Proceedings of 2010 9th International Symposium on Electronics and Telecommunications (ISETC 2010), Nov. 2010, Timişoara, Romania, pp. 373-376
3. [Rus11_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Evaluation of wireless sensor networks localization algorithms", Proceedings of 2011 IEEE 6th International Conference on Intelligent Data Acquisition and Advanced Computing (IDAACS 2011), Sept. 2011, Prague, Czech Republic, pp. 857-862
4. [Rus11_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Evaluation of some node localization algorithms", Proceedings of IEEE 17th International Symposium for Design and and Technology in Electronic Packaging (SIITME 2011), Oct. 2011, Timişoara, Romania, pp. 287-290
5. [Rus11_03] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Performance analysis of wireless sensor network node localization algorithm", Proceedings of 2011 18th IEEE Symposium on Vehicular Technology in Benelux (SCVT 2011,) Nov. 2011, Ghent, Belgium, pp. 1-5
6. [Rus10] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Modeling and simulation of wireless sensor networks for event detection", Proceedings of 2010 International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI 2010), May 2010, Timişoara, Romania, pp. 535-540

In addition to the published papers mentioned above the following paper has been accepted for publication:

[Rus_12] **Ruxandra Ioana Rusnac**, Aurel Gontean and Liviu Crişan, "Wireless Sensor Network Localization Scenario", accepted for publication in the Proceedings of 2012 IEEE 18th International Symposium for Design and Technology in Electronic Packaging (SIITME 2012), pp, Octomber 2012, Alba Iulia, Romania (IEEE Xplore)

REFERENCES

1. [Zur09] Richard Zurawski, "Networked Embedded Systems", CRC Press, chapter 3, pp. 3-6, 2009
2. [Hol05] Holger Karl and Andreas Willig, "Protocols and Architectures for Wireless Sensor Networks" Wiley Interscience, pp. 17-67, 2005
3. [Wal08] Marcus Walchli, Samuel Bissig, Michael Meere and Torseten Braun, "Distributed Event Tracking and Classification in Wireless Sensor Networks", Journal of Internet Engineering, vol. 2, no.1,pp. 117-126, June 2008
4. [Fie06] R.V. Field Jr. and M. Grigoriu, "Optimal design of sensor networks for vehicle detection, classification and monitoring",Elsevier Probabilistic Engineering Mechanics., vol. 21, pp. 305-316, 2006
5. [Vall06] ElMoustapha Ould-Ahmed-Vall and George F. Riley and Bonnie S. Heck, "A Distributed Fault-Tolerant Algoritm for Event Detection Using Heterogeneous Wireless Sensor Networks",Proceedings of 45th IEEE Conference on Decision and Control, USA, 2006.
6. [Ban08] Torsh Banerjee, Bin Xie and Dharma P. Agrawal, "Fault tolerant multiple event detection in a wireless sensor network", Elsevier J.ParallelDistrib.Comput, vol.68, pp. 1222-1234, 2008
7. [Lin09] Chih-Yu Ling, Yu-Chee Tseng and Yung-Chih Liu, "Imprecision-Tolerant Location Management for Object-Tracking in Wireless Sensor Network", British Computer Society The Computer Journal Advance Access, 2009
8. [Fen04] Zao Feng and Leonidas Guibas, "Wireless Sensor Networks- An Information Processing Approach", Elsevier Morgan Kaufmann Series in Networking, pp. 51-55, 2004
9. [Che03] Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee and Chi-Fu Huang, " Location Tracking in Wireless Sensor Newtorks by Mobile Agents and Its Data Fusion Strategies", The British Computer Society, The Computer Journal, vol. 47, no.4, pp. 448-460, 2003
10. [Lee07] Jangwon Lee, Wei Yu and Xonwen Fu, "Energy-efficient target detection in sensor networks using line proxies", International Journal of Communication Systems, Wiley Interscience, vol. 21, pp. 251-275, 2007
11. [Shi08] Kuei-Ping Shih, Sheng-Shih Wang, Huang-Chang Chen and Pao-Hwa Yang, "CollECT: Collaborative event detection and tracking in wireless heterogeneous sensor networks", Elsevier Computer Communications, vol. 31, pp. 3124-3136, 2008
12. [Rus10] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Modeling and simulation of wireless sensor networks for event detection", ICCC-CONTI, May 2010, pp. 535-540
13. [Che05] K.W. Chenung and H.C. So, "A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements", IEEE Transactions on Signal Processing, Vol. 53, No.2, pp. 460-470, February 2005

14. [Ari04] Jagoba Arias et. al., "GPS-less location algorithm for wireless sensor networks", Elsevier Computer Communications, vol. 30, pp. 403-409, 2004
15. [Fin11] Andreas Fink and Helmut Beikirch., "Reliable Radio-based Human Tracking in Hazardous Environments", Wireless Congress- 2011 systems and applications, Munich, Germany, November 2011
16. [Jan11] Adel Javanmard and Andrea Montari, "Localization from Incomplete Noisy Distance Measurements", 2011 IEEE International Symposium on Information Theory, St. Petersburg, Russia, 31 July-5 August 2011, pp. 1584-1588
17. [Rap02] Theodore S. Rappaport, "Wireless Communications- Principles and Practice", Prentice Hall Communication Engineering and Technologies Series, pp. 81-138, 2002
18. [Sea00] Seah, W.K.G. et. al., "Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP) - Survey and challenges", 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE, 2009, pp. 1-5, 2002
19. [Des07] Jasmin Desai and Uf Turelli, "Evaluation Performance of Various Localization Algorithms in Wireless and Sensor Networks", 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio communications, 2007
20. [Rus10_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Maximum Likelihood Estimation Algorithm Evaluation for Wireless Sensor Networks", SYNASC 2010, Sept. 2010, pp. 95-98
21. [Rus10_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Target detection algorithm validation in WSN", ISETC 2010, Nov. 2010, pp. 373-376
22. [Rus11_01] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Evaluation of wireless sensor networks localization algorithms", IDAACS 2011, Sept. 2011, pp. 857-862
23. [Rus11_02] **Ruxandra Ioana Rusnac**, Aurel Gontean, Evaluation of some node localization algorithms", SIITME 2011, Oct. 2011, pp. 287-290
24. [Rus11_03] **Ruxandra Ioana Rusnac**, Aurel Gontean, "Performance analysis of wireless sensor network node localization algorithm", SCVT 2011, Nov. 2011, pp. 1-5
25. [Kua07] Kuang, Xing-Hong and Shao, Hui-He, Distributed localization using mobile beacons in wireless sensor networks, The Journal of China Universites of Posts and Telecommunications, Vol. 18, Issue 4, December 2007
26. [Rus11_01] **Rusnac, Ruxandra-Ioana** and Gontean, Aurel, Performance Alalysis of Wireless Sensor Networks Node Localization Algorithms, 18th IEEE Symposium on Communications and Vehicular Technology in Benelux (SCVT 2011), Ghent, Belgium, 22-23 November 2011
27. [Rus11_02] **Rusnac, Ruxandra-Ioana** and Gontean, Aurel, Evaluation of Wireless sensor Networks Node Localization Algorithms, the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2011), Prague, Czech Republic, 15-17 September 2011, pp. 857-862
28. [Van09] Vandenrberghe , Lieven and Boyd, Stephen, Convex Optimization, Cambridge University Press, 2009, pp. 150-188

29. [Stu99] Sturm, Jos, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optimization Methods and Software, No. 12, 1999, pp. 625-653
30. [Tse07] Tseng, Paul, Second-order cone programming relaxation of sensor network localization, Siam J. Optim, 18, 1, pp. 156-185, 2007
31. [Sri08] Srirangarajan, Seshan, Distributed Sensor Network Localization Using SOCP Relaxation, IEEE Transactions on Wireless Communcations, 7, 12,pp. 4886-4895, 2008
32. [Bao11] Baoshu, Xu, Wenyu, Qu and Wanlei, Zou, A mobile agent-based routing algorithm and some theoretical analysis, Computer Systems Science and Engineering, Vol. 26 , Issue 1 , January 2011, pp: 5-11
33. [Zho10] Zhongwen, Guo, Yuan, Feng , Lu, Hong et al., Efficient and Adaptive Transmission Algorithms for Underwater Acoustic Networks, Computer Systems Science and Engineering, Vol. 25 , Issue 6 , Special Issues, November 2010, pp. 415-425
34. [Shi11] Shirazi, Ghasem Naddafzadeh, Botros Shenouda, Michael, and Lampe, Lutz, Second order cone programming for sensor network localization with anchor position uncertainty, 8th Workshop on Positioning Navigation and Communication (WPNC), pp. 51-55, 2011.
35. [Van11] Vanheel et al. "Automated linear regression tools improve RSSI WSN localization in multipath indoor environment", EURASIP Journal on Wireless Communications and Networking 2011 2011:38.
36. [Rus12] **Rusnac, Ruxandra-Ioana** and Gontean Aurel, "Wireless Sensor Network Real Life Localization", under review IJCT
37. [Nad08] Nadimidi, E.S et. al, "ZigBee-based wireless sensor networks for monitoring animal presence and pasture time in a strip of new grass", Elsevier Computer and Electronics in Agriculture, (61) 2008, pp. 79-87
38. [Tso09] Yu-Tso et. al, "A RSSI-based Algorithm for Indoor Localization Using ZigBee in Wireless Sensor Network", Proceedings of the 15th International Conference on Distributed Multimedia Systems (DMS 2009) , San Francisco, USA, Sep. 2009, pp. 70-75.
39. [Fin09] Fink, Andres, Beikirch, H "RSSI-based indoor localization using antenna diversity and plausibility filter", 2009 6th Workshop on Positioning Navigation and Communication (2009), pp. 159-165
40. [Hyo09] Hyo-Sung Ahn Hyo-Sung Ahn, Wonpil Yu Wonpil Yu, "Environmental-Adaptive RSSI-Based Indoor Localization", IEEE Transactions on Automation Science and Engineering, Volume: 6, Issue: 4, 2009, pp. 1-8
41. [Gol10] Goldoni,Emanuele et. al, "Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4", 2010 European Wireless Conference EW (2010), pp. 71-77
42. [Xbe11] XBee®/XBee-PRO® ZB RF Modules, Datasheet and user manual, Digi International, 2011
43. [Ard11] Arduino Datasheet and User Manual:
    i. http://arduino.cc/en/Main/arduinoBoardDuemilanove
44. [Ers05] ETSI EN 300 328 V1.7.1, 2006
45. [Cox11] Cox, T and Cox, M, "Multidimensional Scaling. Monographs on Statistics and Applied Probabilities", Chapman and Hall, 2001
46. [Oh10]Oh, S et al. , "Sensor Network Localization from Local Connectivity: Performance Analysis for the MDS-map algorithm", ITW 2010

47. [Bis04] Biswas, P, "Semidefinite Programming for Adhoc Sensor Localization", IPSN 2004
48. [Sri07] Srirangarajan, S et. al, "Distributed Sensor Network Localization with Inaccurate Anchor Positions and Noisy Distance Measurements ", ISASSP, 2007
49. [Bis06] Biswas, P et. al., "A Distributed Method for Soving Semnidefinite Programming Arising for Adhoc Wireless Sensor Network Localization", Multiscale Optimization Methods and Applications, Springer Verlang, 2006
50. [Zhe10] Zheng, Y et. al., "Beyond Trilateration: On the Localizability of
51. Wireless Ad-hoc Networks", IEEE ACM Transactions on Networking, 18, 2010, pp.1806-1814
52. [Wan05] Wan, Q et. al., "Mobile Localization Methods on Multidimensional Similarity Analysis", Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, March 2005, Chengdu, China, pp. iv/1081 - iv/1084 Vol. 4
53. [Wan05] Wan, Q et. al., "Mobile Localization Methods on Multidimensional Similarity Analysis", Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, March 2005, Chengdu, China, pp. iv/1081 - iv/1084 Vol. 4
54. [Nic04] Niculescu, D and Nath, Bandri, "Position and orientation in ad hoc networks", Elsevier Ad Hoc Networks 2, 2004, pp. 133-151
55. [You08] Younis, M and Akkaya, K, "Strategies and techniques for node placement in wireless sensor networks: A survey", Elsevier Ad Hoc Networks 6, 2008, pp. 621-655
56. [Lia08] Liao, W-H, "A localization protocol with adaptive power control in wireless sensor networks", Elsevier Computer Communications 31, 2008, pp. 2496-2504
57. [Aro04] Arora, A et, al. , "A line in the sand: a wireless sensor network for
58. target detection, classification, and tracking", Elsevier Computer Networks 46, 2004, pp. 605-634
59. [Yik08] Yick, J et, al. , "Wireless sensor network survey", Elsevier Computer Networks 52, 2008, pp. 2292-2330
60. [Ada11] Adasme, P et, al. , "A Two Stage Stochastic Semidefinite Relaxation for wireless OFDMA Networks", Elsevier Electronic Notes in Discrete Mathematics 37, 2011, pp. 69-74