Precise Localization in Coarse-Grained Localization Algorithms Through Local Learning

Ralf Salomon
University of Rostock
18051 Rostock, Germany
Email: ralf.salomon@etechnik.uni-rostock.de

Abstract—The precise and “cost-efficient” localization is of particular interest in large wireless sensor networks. Approximative algorithms, such as coarse-grained localization with centroid determination, require only very little computational resources (in terms of hardware components) but lack high precision. This paper discusses a few learning schemes, which improve the localization errors by several orders of magnitude at negligible extra costs. Furthermore, these (local) learning schemes makes it possible to apply these localization algorithms to a new class of applications.

I. INTRODUCTION

According to well-established understandings [1], [10], a sensor network consists of a huge number of tiny sensor nodes, which are usually randomly distributed over an area of interest. All sensors measure pre-specified environmental conditions and propagate them by means of wireless radio capabilities to the nearest receivers.

Because of the information processing done at later stages, all sensor nodes are ought to know their (approximate) positions. Since the distribution of a huge number of sensors is typically be done in a rather stochastic process, assigning the location information by handcrafted methods is not feasible at all. Thus, automatic algorithms are required for this task.

The pertinent literature on sensor networks proposes various localization algorithms [4], [5], [7], [16]. It is reasonable to sort these algorithms into two classes: exact and approximative. Exact localization algorithms utilize global methods, such as the global positioning system (GPS), the global system for mobile communication (GSM), and Galileo [9], or derive the exact position by performing some triangulation on the strengths of the incoming signals with respect to known points. In addition, fault-tolerant algorithms, such as linear least squares [14], have been proposed.

The exact methods mentioned above, however, require sensor units with expensive hardware components. They either employ hardware devices, e.g., GPS modules, and/or significant processing capabilities to perform, for example, the required matrix multiplications and inversions. However, the number of sensor nodes might be huge, and thus, the overall costs of sophisticated hardware devices are that high many applications cannot afford. Therefore, the utility of exact localization algorithm is limited to some applications, and are thus not further discussed in this paper.

In order to facilitate the wide-spread dissemination of sensor networks, current designs aim at cheap and small sensor nodes with a long time of operation. Also, typical wireless sensor nodes need to have their own power supply, such as a battery, which favors power-saving algorithms and excludes too high network traffic on which some exact (fine-grained) localization algorithms depend [14]. Under this cost constraint, approximative localization algorithms seem way more appealing. One of these algorithms, also known as coarse-grained localization with centroid determination [4], is briefly reviewed in Section II. This paper focuses on this particular algorithm, since it has received recent interest [2].

Experimental results discussed in Section III indicate unfortunately, coarse-grained localization algorithms are inaccurate by their very nature. Section III shows that the final accuracy depends on various parameters and that in principle, the final error is relatively large. Even though this might be acceptable for some applications, it seems worthwhile to investigate some cost-effective enhancements that yield an accuracy comparable to fine-grained localization algorithms.

For this purpose, Section IV discusses two different learning schemes. They are inspired by neural network learning [13], and indeed reduce the positioning error by several orders of magnitude.

In addition, the employment of a local learning
Section VI concludes with a brief discussion. Terms of processor capabilities and radio modules, but some modifications, which are computationally cheap in [4]. Rather, the goals of this paper is propose and explore algorithm, since this has already been done elsewhere.

Sensor network consisting of tropoid determination [4] assumes a finite two-dimensional space. It should be noted here that the purpose of this paper is not to propose the coarse-grained localization algorithm, since this has already been done elsewhere [4]. Rather, the goals of this paper is propose and explore some modifications, which are computationally cheap in terms of processor capabilities and radio modules but significantly improve the localization accuracy. Finally, Section VI concludes with a brief discussion.

II. BACKGROUND: COARSE-GRAINED LOCALIZATION

The coarse-grained localization algorithm with centroid determination [4] assumes a finite two-dimensional sensor network consisting of $s$ sensor nodes $S_{1 \leq i \leq s}$ and $b$ beacons $B_{1 \leq i \leq b}$. Beacons are a few particular nodes that know their exact $x_B$, $y_B$-coordinates. In the infrastructure case, the beacons are distributed on a grid with distance $d$ between neighbors. The model makes the following four assumptions [2], [4]:

1) all beacons broadcast their messages with perfect circular radio waves and equal transmission range $r$,
2) all sensor nodes $S_i$ perfectly receive all messages from all beacons within transmission range $r$,
3) none of the sensors receives any message from a beacon further away than $r$.
4) no arriving message interferes with any other message arriving at the same sensor.

Figure 1 illustrates an example in which $4 \times 4$ beacons $B_{x,y}$ are distributed over an area $D \times D$ with distance $d$ between neighbors and transmission range $r$.

In the positioning phase, the beacons periodically broadcast their exact positions $x_B$, $y_B$, along with other, potentially essential information. During the constant broadcast time interval $\tau$, the sensor nodes store all incoming messages. At the end of such a period, all sensor nodes approximate their positions $\hat{x}_i$, $\hat{y}_i$ as follows:

$$\hat{x}_i = \frac{1}{n_i} \sum_{k=1}^{n_i} x_{B_k}, \quad \hat{y}_i = \frac{1}{n_i} \sum_{k=1}^{n_i} y_{B_k},$$

with $n_i$ denoting the number of messages arriving at sensor node $i$. Sensor nodes that do not receive any message, i.e., $n_i=0$, are called unknowns and are not further considered. In the general case, the sensor node’s approximated position $\hat{x}_i$, $\hat{y}_i$ differs from its true position $x_i$, $y_i$. The Euclidian distance $e_i(x, y) = \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}$ is called the approximation (or positioning) error $e_i(x, y)$. The average approximation error $\hat{e}$ is given as:

$$\hat{e} = 1/s \sum_{i=1}^{s} e_i(x, y).$$

It is obvious that both approximation errors $e_i(x, y)$ and $\hat{e}$ depend on the actual transmission range $r$. For readability reasons, however, $r$ is not explicitly noted, since it is already subsumed in the approximations $\hat{x}_i$, $\hat{y}_i$. In some cases, it is also worth to consider the maximum error $\hat{e}=\max_i e_i(x_i, y_i)$. Likewise, the transmission range $r$ is not noted explicitly.

III. PROBLEM DESCRIPTION

This section discusses the accuracy with which the sensor nodes derive their approximate positions. For the sake of simplicity, this is done for a simplification, the one-dimensional case, because the mechanism and reasons are identical with the two-dimensional case but are significantly easier to understand.

Even though tolerable for some applications, the resulting average error $\hat{e}$ remains at significant values. Figure 2 shows how both errors $\hat{e}$ and $\hat{e}$ depend on $r$ and that they assume a minimum for $r=0.1875$ for the example $b=5$ and $d=D/4=0.25$. More generally, some
tedious geometrical calculations (their presentation is not in the focus and thus way beyond the scope of this paper) yield an optimal transmission range \( r_{\text{opt}} = 0.75d \) and \( r_{\text{opt}} = 0.86d \) for the one and two-dimensional case [12], respectively.

Furthermore, the resulting error values behave quite sensitive to large transmission ranges \( r > d \) as Fig. 3 clearly illustrates. A serious problem is that the accuracy progressively degrades as the transmission range \( r \) increases. This behavior might be somewhat counter intuitive. But since this paper aims at proposing local learning rules to improve the network's localization accuracy, this paper merely presents a qualitative explanation of this phenomenon. Figure 4 shows a tiny one-dimensional portion of a two-dimensional network as already illustrated in Fig. 1. It can be clearly seen that all sensor nodes within a particular section of the entire network assume the same position, which is the beacon itself in the illustrated example. Therefore, the final accuracy of the estimated position is limited by the size of this section i.e., \( 0.5d \) and \( (2/3)^2d^2 \) in the one and two-dimensional case, respectively. This example also indicates the odd behavior already presented in Fig. 3: In case of a transmission range \( r = 2d + \epsilon \), which is slightly larger than the double distance between neighboring beacons, sensor nodes that are close to beacon \( B_0 \) would derive the position of beacon \( B_1 \), since they would receive messages from beacons \( B_0, B_1, \) and \( B_2 \). The problem continuously increases for further increasing transmission ranges \( r \). A major problem is that the sensor nodes do not have any means by which they can yield a better accuracy, unless they employed high-cost devices.

IV. LEARNING

Learning is well understood in other disciplines, such as neural networks [13], robotics [3], [11], and artificial intelligence in general. The idea of this section is to introduce local learning rules with which the single nodes can fine tune their rough position estimates (obtained by executing the coarse-grained localization algorithm with centroid determination [4] according to Eq. (1)). To this end, this section first summarizes some preliminaries, then introduces the main concepts at a one-dimensional simplification, and finally describes the required generalizations for the two-dimensional case.
A. Preliminaries

The background material presented above suggests to address the following concerns:

1) The choice of the number of beacons (regardless whether one or two-dimensional) determines the different areas and their sizes. In case of an equidistant beacon distribution, the area size is of length \( l = d/2 \). Within each area, all nodes assume the same location (Fig. 4), and thus the minimal values for \( \bar{e} \) and \( \hat{e} \). Once this choice has been made, no further improvements can be attained.

2) The error values \( \bar{e} \) and \( \hat{e} \) react very sensibly to the specifically chosen transmission range \( r \) (Figs. 2 and 3). As a consequence, the network’s localization accuracy progressively degrades for large transmission ranges \( r > d \) (Fig. 3). Unfortunately, this problem significantly limits the algorithm’s robustness with respect to beacon failures and transmission accuracies.

3) The coarse-grained localization algorithm with centroid determination presupposes perfect circular radio waves over a rather large area. This requires that all nodes are in free sight to their neighboring beacons. But this limitation restricts the applicability of the algorithm at hand significantly; the elimination or at least alleviation, by contrast, would open up entirely new application classes.

To significantly broaden the applicability of existing cost-efficient coarse-grained localization algorithms, it would be desirable to at least alleviated if not eliminated the problems mentioned above. To this end, the remainder of this section proposes and investigates two different learning schemes. When using learning, the positioning procedure is split into two stages. In the first stage, the sensor nodes derive a “rough” estimate by performing the coarse-grained localization algorithm with centroid determination (Eq. (1)). In the second stage, then, the sensor nodes apply a learning (or adaptation) step to fine tune their rough estimates. This algorithm variant is therefore called coarse-grained localization with fine tuning through learning, or CGL-FTTL for short.

B. Parallel Learning

Algorithm: Figure 5 shows a small portion of a one-dimensional sensor distribution, which consists of three equally sized sections in which the nodes derive the positions \( p_{i-1}, p_i, \) and \( p_{i+1} \), respectively. The figure also highlights three nodes \( S_{k-1}, S_k, \) and \( S_{k+1} \). It can be observed that within transmission range \( r \), node \( S_K \) has an equal number of nodes with the derived positions \( p_{i-1} \) and \( p_{i+1} \). For the other two nodes, the situation changes. Node \( S_{k+1} \) is one position to the right and has thus one node less with derived position \( p_{i-1} \) but an additional one with position \( p_{i+2} \). For node \( S_{k-1} \), the change is similar but shifted one node position to the left.

The differences in the localizations described above suggest that by providing their derived positions, all nodes can serve as a source for further update information as follows:

- **Parallel learning**: All sensor nodes \( S_i \) broadcast their position estimates with the optimal transmission range \( r \). Afterwards, all sensor nodes apply the coarse-grained localization algorithm with centroid determination (Eq. (1)) to all received messages. In other words, the sensor nodes apply the positioning algorithm twice, once for the beacons’ messages (rough estimate) and once for the former and all received sensor nodes’ messages (fine tuning).

**Results**: Figure 6 clearly shows that learning yields a significant improvement; it decreases the average error

![Fig. 5](image)

![Fig. 6](image)
Fig. 7. This figure shows how learning improves the positioning error of 1000 sensor nodes, which are distributed over a one-dimensional area with $D=1$ and $b=8$ beacons (see also Fig. 6). It can be seen that towards the center, learning drops the positioning error $e_i$ below $10^{-4}$; at the edges, though, the positioning error remains unchanged.

Figures 8 and 9 clearly show how positioning correction (Eq. 4) further improves the location accuracy. Figure 8 also shows that the influence of the chosen transmission range on the average error $\bar{e}$ becomes much less. Furthermore, the minimal average error drops below 0.0002, which is almost two orders of magnitude better than in the original case.

It should be finally noted that the application of the learning rule to the two-dimensional case is straightforward due to the independence of the two coordinate axes $x$ and $y$. A straightforward implementation of the positioning correction algorithm, however, requires

by an order of magnitude for small transmission ranges. In particular, learning reduces the average error from $\bar{e}=0.0178$ to $\bar{e}=0.002$ after one iteration. Similarly, learning reduces the maximal error from $\hat{e}=0.035$ to $\hat{e}=0.0141$.

In addition to the presented results, Fig. 7 shows the positioning error $e_i$ of $s=1000$ sensor nodes, which are distributed over a one-dimensional area of $D=1$ with $b=8$ beacons. It can be seen that towards the center, the positioning error $e_i$ is between 0.0003 and 0.000001 (right-hand-side). In comparison to the case without learning (left-hand-side), the achieved improvement is in the order of one or two orders of magnitude.

Discussion: Figure 7 also indicates that the positioning error remains relatively high at the network’s edges. The results furthermore suggest that the average error $\bar{e}$, as presented in Fig. 6, significantly depends on this effect. The reasons resemble the one already mentioned in Section III: the sensor nodes towards the edges have an asymmetric situation in that they do not receive the same number of messages from the right and left-hand-side. In other words, each sensor unit would receive a total of $n_i=2r\mu$ messages, with $\mu=n/D$ denoting the node density. Towards the edge, the number may shrink to $n_i=r\mu$. The difference between the actual number or received messages $n_i$ and the maximum $2r\mu$ can be directly used to estimate the sensors’ positions:

$$\tilde{x}_i = (n_i - r\mu) \frac{D}{s-1} \quad \text{if} \quad x_{Si} \leq r$$
$$\tilde{x}_i = D - (n_i - r\mu) \frac{D}{s-1} \quad \text{if} \quad x_{Si} \geq D - r \quad (4)$$

Fig. 8. This figure shows how learning and positioning correction (Eq. 4) improve the average error $\bar{e}$. One-dimensional example with $b=8$ beacons over $D=1$ and varying transmission ranges $r$. 

Fig. 9. This figure shows how learning and positioning correction (Eq. 4) improve the average error $\bar{e}$. One-dimensional example with $b=8$ beacons over $D=1$ and varying transmission ranges $r$. 

Discussion: Figure 7 also indicates that the positioning error remains relatively high at the network’s edges. The results furthermore suggest that the average error $\bar{e}$, as presented in Fig. 6, significantly depends on this effect. The reasons resemble the one already mentioned in Section III: the sensor nodes towards the edges have an asymmetric situation in that they do not receive the same number of messages from the right and left-hand-side. In other words, each sensor unit would receive a total of $n_i=2r\mu$ messages, with $\mu=n/D$ denoting the node density. Towards the edge, the number may shrink to $n_i=r\mu$. The difference between the actual number or received messages $n_i$ and the maximum $2r\mu$ can be directly used to estimate the sensors’ positions:

$$\tilde{x}_i = (n_i - r\mu) \frac{D}{s-1} \quad \text{if} \quad x_{Si} \leq r$$
$$\tilde{x}_i = D - (n_i - r\mu) \frac{D}{s-1} \quad \text{if} \quad x_{Si} \geq D - r \quad (4)$$

Figures 8 and 9 clearly show how positioning correction (Eq. 4) further improves the location accuracy. Figure 8 also shows that the influence of the chosen transmission range on the average error $\bar{e}$ becomes much less. Furthermore, the minimal average error drops below 0.0002, which is almost two orders of magnitude better than in the original case.

It should be finally noted that the application of the learning rule to the two-dimensional case is straightforward due to the independence of the two coordinate axes $x$ and $y$. A straightforward implementation of the positioning correction algorithm, however, requires
memory space in the order of $r\mu$ entries per coordinate axis and additional CPU resources.

C. Sequential Learning

**Algorithm:** The parallel learning algorithm described in the previous subsection might not be suitable for all application types. For such cases, this subsection proposes a sequential variant as an alternative. The algorithm makes the following assumptions:

1) Sensor nodes are able to distribute particular positioning messages over a small transmission range

$$r_s = D/n + \epsilon = 1/\mu + \epsilon,$$  \hspace{1cm} (5)

which has to be slightly larger than the average distance between neighboring sensor nodes $D/n$

2) All sensor nodes store positioning messages over some time period $t_s$,

3) Sensor nodes forward only those positioning messages they have not seen during the current time period $t_s$, i.e., which they do not have in their own memory (in order to avoid loops).

4) The algorithm views the sensor nodes between two neighboring beacons $B_i$ and $B_{i+1}$ as a chain.

In the learning stage, one particular sensor node $S_i$ per chain communicates a specific positioning message with a unique code and a hop count $h_{S_i}=0$. This message then ripples through the chain towards both ends in the following way: The receiving sensor nodes increase the hop count by one and then themselves forward the message to the neighbors. This process terminates when a positioning message reaches the beacons $B_i$ and $B_{i+1}$.

Once the beacons have received the message (with the same message id but potentially different hop counts), they broadcast the message id and the final hop count along with their own position. All receiving sensor nodes can then estimate (reconstruct) their relative positions as follows:

$$\bar{x}_i = x_{B_i} + \frac{h_{B_i} - h_i}{h_{B_{i+1}} + h_{B_i}} (x_{B_{i+1}} - x_{B_i}),$$  \hspace{1cm} (6)

with $h_{B_i}$ and $h_{B_{i+1}}$ denoting the hop counts of beacons $B_i$ and $B_{i+1}$, respectively. With this scheme, the sensor nodes can derive two possible positions. The result is ambiguous due to lack of orientation information; the sensors transmit their data in perfect circular radio waves into all directions equally well. Thus, this last step has to be repeated once: Another sensor node $S_{i\neq k}$ has to initiate a second positioning message. Since then all nodes derive another set of two possible positions, the intersection is the true physical location.

**Results:** Performance figures are not presented here, because they would end up at error values $\bar{e}=0$, since the sensor nodes are regularly distributed on a grid [2]. In case the sensor nodes are randomly distributed, the final error approaches the standard deviation of the chosen error model.

**Discussion:** This learning algorithm has very little computational demands. The sensor nodes have to store and send two particular positioning messages (i.e., the messages’ unique ids and hop counts). Since the messages have to be send only to the adjacent sensor nodes, the required transmission range $r_s$ is very small, which will not affect their energy resources too much. Due to the algorithm’s nature, however, this fine tuning takes some time, since the information “ripples” along the complete chain, i.e., up to $s/(b-1)$ nodes.

The approach that positioning messages be send only between adjacent sensor nodes (rather than adjacent beacons) relaxes the assumptions significantly. As discussed in the background material, the coarse-grained
localization algorithm with centroid determination assumes perfect global transmission conditions between beacons. By contrast, the proposed local learning rules require only perfect local transmission conditions between neighboring sensor nodes.

The algorithm still functions well, if the small transmission range $r_s$ has to be increased for any reason. If, for example, a transmission range twice as large is required (due to local inhomogeneities or local disturbances), two neighboring sensor nodes will be storing the same hop count. The estimation accuracy thus degrades resulting in an average error $\bar{e} \approx D/n = 1/\mu$ in the order of the distance between neighboring sensor nodes.

The generalization to two dimensions is straightforward. With respect to the four enclosing beacons $B_{i,j}$, $B_{i,j+1}$, $B_{i+1,j}$, and $B_{i+1,j+1}$, each sensor node can estimate the four distances and can thus approximate their true position.

V. ACCURATE LOCALIZATION AND OBSTACLES

As has been discussed in the background material, the assumption of perfect circular radio waves require that all sensor nodes are in free sight to the nearest beacons. This in turn, excludes any obstacles, since they would disturb or even completely block the transmission of the radio waves. The very same problem also applies to other methods, such as distance estimation and triangulation. One solution might be the utilization of further beacons, which however, results in higher costs.

The proposed local learning rules, on the other hand, provide a natural way to cope with the presence of obstacles. Figure 11 illustrates an example (a very small portion of a large sensor network), in which an obstacle prevents several sensor nodes from (directly) receiving messages from the four beacons. When employing sequential learning, as described in Subsection IV-C, all sensor nodes receive messages with varying hop counts, depending on what they receive (and what not), they can yield at least a reasonable position estimate. Without learning, coarse-grained localization algorithms could not be applied to such applications.

VI. CONCLUSION

This paper has first reviewed a well-known localization algorithm, known as coarse-grained localization with centroid determination, as well as some basic properties. The paper has then shown that the attainable accuracy is inherently limited and that it reacts very sensibly to the chosen transmission range $r$. To tackle these limitations and to thus significantly broaden the algorithms applicability, this paper has both proposed and validated two different learning rules, called parallel and sequential learning. These learning schemes consist of two stages, in which they first derive rough estimates and then fine tune them. This fine tuning enables the sensor nodes to improve their position estimates significantly: the improvement is up to two orders of magnitude.

Despite an improved accuracy, the proposed learning schemes feature the following advantages: (1) since the learning scheme involves only neighboring beacons $B_i$ and $B_{i+1}$, it scales perfectly over the number $b$ of beacons; (2) due to the learning scheme, the average error $\bar{e}$ reacts much less sensitive to the chosen transmission range $r$; and (3) the incorporation of sequential learning allows for the consideration of obstacles, which would make other methods fail.

A comparison of the two proposed learning schemes reveal that parallel learning requires more energy (larger transmission ranges of the sensor nodes) and that sequential learning requires more time (due to the rippling nature). In addition, sequential learning achieves the same accuracy in all regions, and thus yields the lowest average error overall; that is sequential learning does not
exhibit an increased error at the area’s ends as all the other schemes do.

This paper has focused on the introduction of complementary concepts and their experimental validation. Future research will be devoted to thorough analyses, which will be aiming at providing analytical descriptions, especially of the dependencies between transmission range $r$ and achievable average errors $\bar{e}$. Furthermore, future research will thoroughly investigate a case study, which features several obstacles.

ACKNOWLEDGMENT

The author would like to thank Dirk Timmermann for his continuous support, as well as Jan Blumenthal and Frank Reichenbach for encouraging discussions.

REFERENCES