Localization in Ad-Hoc Sensor Networks

Reino Virrankoski

Control Engineering Laboratory, Helsinki University of Technology

E-mail: reino.virrankoski@hut.fi
Student No. 54062J

Abstract

Localization of sensor nodes is essential component in many ad-hoc sensor network applications. This paper presents an overview to existing localization techniques.

Keywords: localization, positioning, ad-hoc networks, sensor networks

1. Introduction

Knowledge about the locations of individual nodes can be a useful primitive in many routing protocols, it can answer the questions on the network coverage and it can assist the group querying of sensors. When reporting about the origins of events in network area or when network is used to the tracking of moving target, the node location information is essential. Location information is also fundamental part in many practical sensor network solutions, like smart highway, smart shopping mall, smart battlefield, industrial applications in logistics and robotics, or in medical applications, where sensors implemented in human body communicate with other devices.

The area of ad-hoc sensor networks is currently under rapid development and very intensive research, and there exists several methods developed for localization. Method types described here are localization with beacons, localization with moving beacon and beacon-free localization. In addition to that, there is a short overview to other possible localization methods.

In many localization applications satellite based positioning sounds like easiest solution to the problem. Currently GPS [1] is well-known and well functioning system, and in the future there is probably alternatives and competition in the satellite-based positioning, if European Galileo [2] and Russian GLONASS [3, 4] get completed. Anyway, in the case of sensor networks satellite-based positioning is not the straightforward solution. The line of sight problems make indoor positioning by satellite-based positioning system impossible, and cause also remarkable problems in many outdoor positioning cases, if there exists obstacles like big buildings or leaves in the trees blocking the line of sight. In addition to that, the power consumption of satellite-based positioning will reduce the battery life of sensor nodes, the production cost of satellite positioning device could dramatically increase the production cost of single node and the size of satellite positioning device and its antenna remarkably increases the size of sensor node – nodes should be small and cheap.
2. Some network parameters

2.1 Node density

Node density $D$ is defined to be a number of nodes in unit area. For $N$ nodes deployed in circular area $A$

$$D = \frac{N}{A}. \quad (1)$$

Thus we get an equation to the radius $L$ of circular field [5]:

$$L = \sqrt{\frac{A}{\pi}} = \sqrt{\frac{N}{D\pi}}. \quad (2)$$

2.2 Network connectivity

The number of other nodes in single node transmission range is called the network connectivity. In a circular field the probability of node having $d$ neighbours can be expressed as binomial probability

$$P(d) = \binom{N-1}{d} P^d (1 - P)^{N-d-1}, \quad (3)$$

where $R$ is the node transmission range, and the probability $P_R$ that the node is within transmission range $R$ from other node is

$$P_R = \frac{\pi R^2}{A} = \frac{D\pi R^2}{N}. \quad (4)$$

As $N$ approaches infinity, the binomial distribution (3) converges to a Poisson distribution

$$P(d) = \frac{\lambda^d}{d!} e^{-\lambda}, \quad (5)$$

where $\lambda = NP_R$. From (5) the probability of node having $n$ or more neighbours is [5]

$$P(d \geq n) = 1 - \sum_{i=0}^{n-1} P(i). \quad (6)$$

3. Localization with beacons

In the network, there are some nodes that have knowledge about their location. Those nodes are called anchors or beacons. Those nodes can get the location awareness by using some additional positioning system like GPS, or the location information can be added to
those nodes when initially configuring the network. Rest of the nodes that don’t have knowledge about their location at the beginning are called *unknowns*.

![Figure 1](image.png)

*Figure 1.* Initial configuration of sensor network. Black points remark beacons, white points unknowns.

To define its location, every node should have capacity to perform *ranging*, in other words estimate its distances from the neighbouring nodes. In the first phase, nodes perform ranging and define preliminary estimates of their location based on ranging information. In the second phase, the location estimates are refined to get better accuracy.

There exist several ranging techniques. Most generally used ones are based on RF time of flight [6], ultrasound [7], ultra wide band [8] or laser time of flight [9]. When using RF time of flight, any additional transmitters for ranging are not needed, but there exist similar type of problems with signal fading and multipath, than in the case of network based cellular phone localization. Using ultrasound is promising, because ultrasound transmitters and receivers are small, cheap, easy to add to the nodes, and measuring the time of arrival (ToA) of ultrasound or time difference of arrival (TDOA) between simultaneously transmitted RF and ultrasound signals give quite accurate results. Ultrasound also suffers some multipath effects, but those effects are easier to detect. ToA measurements use the first pulse received ensuring that the shortest path reading is observed. Thus, in the case of sensor nodes one usually gets the straight line between two nodes, and reflected pulses from nodes that do not have direct line of sight are filtered out [7].

It is possible to achieve very high ranging accuracy by using laser, but in the case of sensor nodes laser might be difficult to implement, it is very sensitive to the line of sight and weather conditions and also the price and size could cause problems. Anyway, in some solutions where network is usually used in good line of sight conditions and very high ranging accuracy is needed, laser is one possible alternative.

After ranging, nodes compute initial estimates about their location based on ranging information and beacon locations. Following phase is location refinement. There are different iterative methods how nodes can change information about their location estimates with their neighbours, and iteratively by round by round get their location estimates better. Then it is possible to use some optimization method to get best possible location estimate [10].
When localization is done with beacons, one important factor in the network behaviour is that how big fraction of nodes are beacons. If the percentage of beacons is high, it is easier to perform localization, but in that case also the production costs and energy consumption of the network are higher, because beacons are more expensive and more power-hungry than unknowns.

In two dimensions unknown can estimate its location if there are three or more beacons in its range. When unknown gets estimate for its location, it becomes beacon and other unknowns can use it for their location estimation. There are different possibilities how to construct the localization algorithm and how to divide the computation between nodes. All computation can be done centralized in one node, it can be totally distributed or something between those two possibilities. Crucial challenges in localization with beacons are how to make localization algorithms as robust as possible by using as small percentage of beacons as possible with as low consumption of power and radio recourses as possible. In the section 7 one localization system with beacons [10] is described more detail.

4. Localization with moving observer

4.1 System description

In some cases, remarkable savings in power consumption can be achieved by using moving beacon often called observer. In the simplest case, sensor nodes are static and observer moves on same fixed path. In practical solutions, for example buses, trains, elevators, robots, forklifts in storehouses, helicopters or armoured vehicles in battlefield or regularly moving persons in hospital wards could carry an observer. Because of the predictability of the observer’s motion, sensor nodes can be in sleep mode until the observer arrives, then wake up, send their data and return back to the sleep mode. When also data transfer is initiated by observer, and collision avoidance etc. are in observer’s responsibility, this leads to the remarkable power savings in sensor nodes [11].

Observe the network consisting $N$ identical nodes having same maximum transmission range $R_{\text{max}}$ and same transmission time (time one sensor need to send all its data to observer) $T$. Assume that the observer is moving by constant speed $v$. To get data transmitted, sensor should stay at least time $T$ in the range of observer. If the observer moves out of the range before, this can lead to the data loss called outage. To ensure the successful data collection in the sense of geometry, every sensor should be located to the distance $R$ or less from the observer path [11].

![Figure 2. Geometric situation.](image-url)
When observer velocity $v$ and maximum distance $R$ between sensor nodes and observer path are known, we get the rule

$$R_{\text{max}} \geq \sqrt{R^2 + (vT/2)^2} \tag{7}$$

for node transmission range $R_{\text{max}}$.

### 4.2 Organizing the queuing

Next issue what needs to be organized is queuing. The assumption that sensors are randomly distributed leads in some places into observer path situations, where observer receives simultaneously several requests to send from sensors, and each sensor has only limited time to send it data to observer. Thus, some kind of queuing system based on arrival process and distribution of waiting times is needed to avoid outages [11].

![Figure 3.1](image1.png)

**Figure 3.1.** When the node density in the observer area is low, there is no problem with the simultaneous requests to send.

![Figure 3.2](image2.png)

**Figure 3.2.** When the observer enters to the area where node density is higher, queuing system is needed.

Assume that $N$ nodes are independently and uniformly distributed over area $A$. In time $t$ the observer travels a distance $vt$. Then nodes in area $2Rvt$ that was previously out of the range come within range. If $p(t)$ is the interarrival probability density function, then probability $P(t)$ that at least one node enters within range in time $t$ is
\[ P(t) = \int_0^t p(x)dx . \]  

(8)

On the other hand, \( P(t) \) is equal to 1 - probability that no node enters within range on time \( t \):

\[ P(t) = 1 - \left( \frac{A - 2Rvt}{A} \right)^N . \]  

(9)

We get the probability density function of interarrival times by setting (8) and (9) equal and taking derivatives from both sides with respect to \( t \) [11]:

\[ p(t) = \frac{2NRv(A - 2Rvt)^{N-1}}{A^N} , \quad 0 \leq t \leq \frac{A}{2Rv} . \]  

(10)

The upper limit to \( t \) in (10) comes from the fact that when observer is moving by velocity \( v \), its range covers the whole network area in time \( A/2Rv \).

It is important to know whether the sensor will stay in the range long enough that it makes sense to start communicating with observer. Maximum waiting time of the sensor is the maximum time what sensor can wait after observer arrives to its communication range before starting communication with observer that there is still enough time to send all the data what sensor should send. If the sensor distance from the path of the observer is \( d \), where \( 0 \leq d \leq R \), the maximum waiting time for that sensor node is [11]

\[ t_w = \frac{2\sqrt{R_{\text{max}}^2 - d^2} - vT}{v} . \]  

(11)

Figure 4. Maximum waiting time.

Because nodes are independently and uniformly distributed to the network area \( A \), also node distances \( d \) from observer path is uniformly distributed from 0 to \( R \). Thus, the probability density function of waiting times can be obtained from (11):
\[ p_{t_w}(t_w) = \frac{p_d(d)}{\frac{1}{3} \tau_x(d)} = \frac{v^2(t_w + T)}{4R \sqrt{R_{\text{max}}^2 - \frac{v^2(t_w + T)^2}{4}}} \]  

(12)

where

\[ \frac{2 \sqrt{R_{\text{max}}^2 - R^2}}{v} - T \leq t_w \leq \frac{2R_{\text{max}}}{v} - T. \]  

(13)

In the network initialization, the waiting times of each sensor should be found out by exchanging information between sensors and observer. After that the queuing when observer moves entire cycle through the network area \( A \) can be handled by using distributions (10) and (12) [11].

The quality criteria for data processing in network can be determined by maximum allowable outage. In many applications outage is not acceptable at all. In that case one should define minimum separation, a minimum distance needed between each of the randomly distributed nodes that ensure that at least geometry doesn’t cause any data loss in the network. In [11] it is shown that the condition

\[ d \geq \sqrt{(2R)^2 + (vT)^2} \]  

(14)

ensures that after one sensor enters to the range of the observer there is at least time gap \( T \) before next sensor enters to the observer range.

4.3 Network initialization and use

The initialization and use of wireless sensor network, where moving observer is used can be divided to three main phases: startup phase, steady phase and failure detection. Each sensor has a unique address, but at the beginning the observer knows nothing about the location of individual sensors [11].

There are two cycles in the startup phase. In the first cycle, the observer goes on its regular path and sends beacon signal by same strength at which sensor nodes transmit. Sensors listen channel and each of them is able to measure how often the observer comes within sensors range and how long it stays within range [11].

In the second cycle of the startup phase, observer travels on its regular bath and broadcasts beacon signal. When sensor hears the signal, it responds with a Request To Send (RTS) packet containing its address. When the observer hears particular sensors RTS, it stops broadcasting a beacon signal, and sends Clear To Send (CTS) packet addressed to the sensor. After receiving that, sensor sends a data packet, which contains information about the parameters measured during first cycle. After receiving this packet, observer sends Acknowledgement (ACK) to the sensor node [11].

In practice, both of the cycles in startup phase should be repeated several times to get information from all sensors, which are in the range of observer path. After that observer use the information collected from sensors and information about its own positions and path to calculate the positions of different sensors and formulate the queuing. Observer
initiates the communication using a wake signal to sensors that it knows to be within range. When there are several sensors simultaneously in the range, observer set higher priority to sensors that can wait less [11].

When observer can’t get response to wake calls, it detects failure. By following those failures it is possible to estimate when network is not longer be able to gather sufficient data without reparations [11].

4.4 More complex network structures

In the sensor network with moving observer described in [11] it was assumed that sensors are static and all of them are located from one hop distance to observer. In the case of multihop distances and moving sensors the system becomes more complex and the algorithms should be developed further.

5. Beacon-free localization

5.1 Benefits of beacon-freeness

The use of beacons increase the price of sensor network and even if it is assumed that sensors are scattered randomly at the beginning, in practice there might always exist larger areas without beacons in the network. Furthermore, error cumulation is one problem in localization algorithms, where also unknowns act as beacons after defining their own position. One possibility how to reduce the network price and how to try to avoid problems detected in localization with beacons is beacon-free localization.

In beacon-free localization relative map of sensor locations is computed. In some cases knowing the relative positions of the nodes compared to each other is enough, and if absolute map of the network is needed, the transformation from one coordinate system to another in two dimensions is determined, if the coordinates of three separate points are known. Respectively in three dimensions the transformation is determined if the coordinates of four separate points are known.

Like in the case of localization with beacons, also now nodes have ability to perform ranging. In other words nodes can measure their distances form those neighbouring nodes, which are located on ranging distance. After that the network can be handled as one coordinate system, or it can first be divided to smaller clusters which form subcoordinate systems, and finally connect the result.

5.2 Multidimensional scaling

Multidimensional scaling (MDS) is a set of data analysis techniques that displays distance-like data as geometrical picture. It is mostly used in psychophysics and cognition science [12], but it is also possible to apply the method to localization [13]. There are several types of MDS techniques. They can be classified according to data (qualitative or quantitative), number of similarity matrices and the nature of the MDS model. The simplest case is classical metric MDS presented here.
The goal of metric MDS is to find a configuration of points in a multidimensional space such that the inter-point distances are related to a provided proximity by some transformation. If the proximity data is generated with Euclidian distances, then classical metric MDS can exactly recreate the configuration of points.

Mark $p_{ij}$ the proximity measure between objects $i$ and $j$, and the Euclidean distance between those points in $n$-dimensional space is

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}.$$  (15)

If the geometrical model fits perfectly to measured data, the Euclidean distances are related to proximities by transformation

$$d_{ij} = a = b(p_{ij}).$$  (16)

One can define

$$I(P) = G + E,$$  (17)

where $P$ is the matrix of measurements, $G$ is the matrix of real distances and $E$ is the matrix of errors. $G$ is a function of the coordinates $X$. The goal of classical metric MDS is to compute the coordinate matrix $X$ such that the sum of squares of $E$ is minimized [13]. Any point can be selected as the origin, but double-centering is recommended [14]. It is suspected that proximities might have random errors then choosing the centroid of the space will tend to minimize the errors as they cancel each other out [13]. When building a relative map, only a relative location of origin is needed.

Let $D$ be the matrix of measured distances after possible transformations performed. $D$ is converted to double-centered matrix $B$ by conversion

$$B = -\frac{1}{2} (I - \frac{1}{N}U)D^2 (I - \frac{1}{N}U) = -\frac{1}{2} ZD^2 Z,$$  (18)

where $U$ is $n \times n$ matrix consisting entirely of 1s, $I$ is identity matrix and $D^2$ is matrix, where matrix $D$ elements are squared $d_{ij}^2$. In these type of relative map definition, $B$ is symmetric square matrix, $B = B^T$ [12, 13].

In linear algebra, it can be shown that the decomposition of quadratic matrix $A$ into the product $LHVL$ where $L$ is lower triangular, $U$ is upper triangular and $H$ is diagonal matrix is unique. Here scalar products are symmetric. Thus

$$B = LHU = B^T = (LHU)^T = U^T H^T L^T.$$  (19)

In addition to that, $L = U^T , U = L^T$ and $H = H^T$. For symmetric matrix $S$

$$S = LH L^T = U^T H U.$$  (20)
If \( H \) is splitted into two matrices, then

\[
B = LHL^T = LH \frac{1}{2} H^\frac{1}{2} L^T = \left( LH \frac{1}{2} \right)^T \left( LH \frac{1}{2} \right) = XX^T. \tag{21}
\]

From (21) we get coordinate matrix [13]

\[
X = LH \frac{1}{2}. \tag{22}
\]

Factoring \( B \) into \( LHL^T \) is called orthogonal diagonalization, and it is always possible for a square symmetric matrix [13, 15]. One dexterous factoring technique is singular value decomposition (SVD).

Given an \( m \times n \) matrix \( A \) where without loss of generality \( m \geq n \) and \( \text{rank}(A) = r \), the SVD of \( A \) is defined as

\[
A = UGV^T, \tag{23}
\]

where \( U^T U = V^T V = I_n \), and \( G \) is a diagonal matrix \( \left( \sigma_1, \ldots, \sigma_n \right) \), where \( \sigma_i > 0 \) for \( 1 \leq i \leq r \) and \( \sigma_j = 0 \) for \( j = r + 1 \). The first \( r \) columns of the orthogonal matrices \( U \) and \( V \) define the orthogonal eigenvectors associated with \( r \) nonzero eigenvalues of \( AA^T \) and \( A^T A \) respectively. The orthogonalization of a square symmetric matrix is a special case of SVD. Thus taking the SVD of \( B \) will give us the solution we want [13, 15, 16].

For example tabulated distances between some Finnish places are following [17]

<table>
<thead>
<tr>
<th>Helsinki</th>
<th>Jyväskylä</th>
<th>Kilpisjärvi</th>
<th>Kuusamo</th>
<th>Oulu</th>
<th>Rovaniemi</th>
<th>Tampere</th>
<th>Joensuu</th>
<th>Turku</th>
<th>Vaasa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helsinki</td>
<td>0</td>
<td>271</td>
<td>1209</td>
<td>803</td>
<td>611</td>
<td>834</td>
<td>173</td>
<td>438</td>
<td>165</td>
</tr>
<tr>
<td>Jyväskylä</td>
<td>271</td>
<td>0</td>
<td>937</td>
<td>552</td>
<td>340</td>
<td>562</td>
<td>151</td>
<td>244</td>
<td>305</td>
</tr>
<tr>
<td>Kilpisjärvi</td>
<td>1209</td>
<td>937</td>
<td>0</td>
<td>625</td>
<td>597</td>
<td>433</td>
<td>1088</td>
<td>983</td>
<td>1233</td>
</tr>
<tr>
<td>Kuusamo</td>
<td>803</td>
<td>552</td>
<td>625</td>
<td>0</td>
<td>212</td>
<td>192</td>
<td>703</td>
<td>458</td>
<td>847</td>
</tr>
<tr>
<td>Oulu</td>
<td>611</td>
<td>340</td>
<td>597</td>
<td>212</td>
<td>0</td>
<td>222</td>
<td>491</td>
<td>396</td>
<td>635</td>
</tr>
<tr>
<td>Rovaniemi</td>
<td>834</td>
<td>562</td>
<td>433</td>
<td>192</td>
<td>0</td>
<td>714</td>
<td>550</td>
<td>585</td>
<td>858</td>
</tr>
<tr>
<td>Tampere</td>
<td>173</td>
<td>151</td>
<td>1088</td>
<td>703</td>
<td>491</td>
<td>714</td>
<td>0</td>
<td>395</td>
<td>153</td>
</tr>
<tr>
<td>Joensuu</td>
<td>438</td>
<td>244</td>
<td>983</td>
<td>458</td>
<td>395</td>
<td>550</td>
<td>395</td>
<td>0</td>
<td>549</td>
</tr>
<tr>
<td>Turku</td>
<td>165</td>
<td>305</td>
<td>1233</td>
<td>847</td>
<td>635</td>
<td>858</td>
<td>153</td>
<td>549</td>
<td>0</td>
</tr>
<tr>
<td>Vaasa</td>
<td>420</td>
<td>283</td>
<td>917</td>
<td>531</td>
<td>319</td>
<td>542</td>
<td>493</td>
<td>332</td>
<td>0</td>
</tr>
</tbody>
</table>

By dividing these distances by 10, we get measured distances matrix \( D \):

\[
D = \begin{bmatrix}
0    & 27.1 & 120.9 & 80.3 & 61.1 & 83.4 & 17.3 & 43.8 & 16.5 & 42  \\
27.1 & 0    & 93.7  & 55.2 & 34   & 56.2 & 15.1 & 24.4 & 30.5 & 28.3 \\
120.9 & 93.7 & 0    & 62.5 & 59.7 & 43.3 & 108.8 & 98.3 & 123.3 & 91.7 \\
80.3  & 55.2 & 62.5  & 0    & 21.2 & 19.2 & 70.3  & 45.8 & 84.7  & 53.1 \\
61.1  & 34   & 59.7  & 21.2 & 0    & 22.2 & 49.1  & 39.5 & 63.5  & 31.9 \\
83.4  & 56.2 & 43.3  & 19.2 & 22.2 & 0    & 71.4  & 55   & 85.8  & 54.2 \\
17.3  & 15.1 & 108.8 & 70.3 & 49.1 & 71.4 & 0     & 39.5 & 15.3  & 24.4 \\
43.8  & 24.4 & 98.3  & 45.8 & 39.5 & 55   & 0     & 54.9 & 49.3  & 33.2 \\
16.5  & 30.5 & 123.3 & 84.7 & 63.5 & 85.8 & 15.3  & 0    & 33.2  & 0  \\
42    & 28.3 & 91.7  & 53.1 & 31.9 & 54.2 & 24.4  & 49.3 & 33.2  & 0
\end{bmatrix} \tag{24}
From (18) we get

\[ Z = (I - \frac{1}{N} U) \]

\[
\begin{bmatrix}
0.9 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & 0.9 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & 0.9 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
-0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 \\
\end{bmatrix} \] \quad (25)

Using (18), (24) and (25) we get double centered matrix \( B \):

\[ B = -\frac{1}{2} Z D^2 Z \]

\[
\begin{bmatrix}
2028 & 795.2 & -3224.8 & -1470.5 & -754.9 & -1723.2 & 1401.5 & 524.9 & 2021 & 402.7 \\
795.2 & 296.8 & -1771.8 & -635.6 & -331.9 & -690.3 & 571.6 & 320.9 & 826.4 & 18.7 \\
-3224.8 & -1771.8 & 6139.2 & 1856 & 1385.3 & 2872.7 & -2312 & -1291.7 & -3388.8 & -864.1 \\
-1470.5 & -635.6 & 1856 & 1479.1 & 612.6 & 1295.8 & -1194.3 & 160.9 & -1704.4 & -399.5 \\
-754.9 & -331.9 & 1385.3 & 612.6 & 195.5 & 591.8 & -570.5 & -212.3 & -775.3 & -140.3 \\
-1723.2 & -690.3 & 2872.7 & 1295.8 & 591.8 & 1481.1 & -1271.3 & -301.8 & -1797.2 & -457.5 \\
1401.5 & 571.6 & -2312 & -1194.3 & -570.5 & -1271.3 & 1074.3 & 227.2 & 1563.2 & 510.2 \\
524.9 & 320.9 & -1291.7 & 160.9 & -212.3 & -301.8 & 227.2 & 940.2 & 106.2 & -474.4 \\
2021 & 826.4 & -3388.8 & -1704.4 & -775.3 & -1797.2 & 1563.2 & 106.2 & 2286.2 & 862.7 \\
402.7 & 18.7 & -864.1 & -399.5 & -140.3 & -457.5 & 510.2 & -474.4 & 862.7 & 541.5 \\
\end{bmatrix} \] \quad (26)

When computing singular value decomposition to \( B \) as presented in (23), we get

\[ B = USV^T. \] \quad (27)

Because \( B \) is symmetric square matrix, the SVD becomes

\[ B = USV^T = USU^T, \] \quad (28)

where

\[
S = \begin{bmatrix}
1421 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 196.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 26.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 23.2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 13.3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 9.5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.00 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.00 & 0 \\
\end{bmatrix} \] \quad (29)
From (21) and (22) we get solution for coordinate matrix $X$:

$$ B = USU^T = \left(US^\perp\right)^TUS^\perp = XX^T $$

$$ \Leftrightarrow X = US^\perp. \quad (30) $$

When setting values of $U$ and $S$ from (28) and (29) we get

$$ X = \begin{bmatrix}
-43.81 & -0.18 & -8.10 & -0.20 & -0.10 & 1.49 & 6.48 & -1.73 & 0.22 & 0.00 \\
-17.01 & -1.15 & -9.17 & -5.62 & 2.38 & 6.70 & -1.99 & 0.37 & -0.35 & 0.00 \\
75.78 & 17.52 & -11.45 & 6.49 & 0.70 & 0.15 & 0.83 & 0.49 & -0.00 & 0.00 \\
31.25 & -19.15 & 13.91 & 3.55 & -5.84 & 4.16 & 2.34 & 0.52 & -0.07 & 0.00 \\
17.43 & 1.14 & 1.93 & -9.44 & -0.73 & -6.08 & 3.19 & 1.70 & -0.14 & 0.00 \\
38.47 & -2.96 & 2.29 & -5.96 & -1.53 & -0.78 & -3.85 & -3.22 & 0.19 & 0.00 \\
-32.30 & 6.15 & -1.97 & -0.58 & -5.31 & 0.89 & -3.45 & 2.53 & 0.38 & 0.00 \\
-10.79 & -29.20 & -6.52 & 4.96 & 5.94 & -3.92 & -1.91 & 0.42 & 0.02 & 0.00 \\
-46.76 & 12.70 & 2.91 & 5.66 & -5.34 & -3.72 & -1.58 & -1.29 & -0.33 & 0.00 \\
-12.27 & 15.13 & 16.17 & 1.15 & 9.84 & 1.11 & -0.05 & 0.21 & 0.08 & 0.00
\end{bmatrix}. \quad (31) $$

The last operation scales the coordinates by the importance of the derived dimensions as represented by the eigenvalues of $D$. SVD will always extract the most significant dimensions first in (29)-(30) [13]. The order of the cities in (31) is same than it was in the first distance matrix (24). Two first columns give the coordinates in two dimensions.

![Figure 5](image-url)  
**Figure 5.** Coordinates of Finnish places in relative coordinate system given by the first two columns of (31).
When applying multidimensional scaling in sensor localization, there exist two main weaknesses. First is that the bigger error in one measurement will affect to all position estimates and second one is that the system robustness in the case of not so isotropic square type or circular type of network area might be weaker. One solution how to alleviate those problems could be that the network is first divided into smaller clusters by some reasonable algorithm, after that multidimensional scaling is performed in each of the clusters and then the results are combined. In principle the multidimensional scaling method presented here in two dimensions can be generalized in three dimensions, but also the robustness of that case requires more investigation.

5.3 Mass-spring model analogy

Observe mass-spring model, where mass points are connected to each other by springs. In the lowest energy stage called relaxation, the springs between mass points are neither stretched nor compressed, and hence the resulting force is zero. One can think the analogy between sensor node location estimates, ranging measurements and mass spring model. Assume that we have some kind of rough initial estimates to the node positions. Nodes can perform ranging and change information about position estimates and ranging results with each other. The distance between two neighbouring nodes can be computed from location estimates and it can be measured. If the measurements and position estimates are absolutely correct, both have same value and hence difference between them is zero. This situation is analog to mass-spring model relaxation stage. If those two distance estimates are not equal, the difference between them is analog to the stretch or compression of the spring between two mass points. In the case of mass spring model we get the resultant
force by computing the vector sum over all forces caused by single connections, and then we can reduce the energy stage of the model by moving the masspoint to the direction of that force. We can use the analogy to the errors of the distance estimates of single neighbour nodes and find the direction in which we can change the localization estimate to reduce the error [18].

Let \( r_{ij} \) be the measured distance between nodes \( i \) and \( j \), and \( d_{ij} \) be the distance computed on the base of the position estimates of \( i \) and \( j \). Then the force is given by

\[
\vec{F}_{ij} = (d_{ij} - r_{ij}) \vec{u}_{ij},
\]  
(32)

where \( \vec{u}_{ij} \) is the unit vector directed from node \( i \) to node \( j \). When computing the distances and measuring the ranges to all of the nodes in node \( i \) one-hop environment, we get the resultant force

\[
\vec{F}_i = \sum_j \vec{F}_{ij}.
\]  
(33)

Because \( \vec{u}_{ij} \) is unit vector in (32) the length of \( \vec{F}_{ij} \),

\[
|\vec{F}_{ij}| = |d_{ij} - r_{ij}|.
\]  
(34)

The total error in node \( i \), analog to total energy of masspoint \( i \) in mass spring model is

\[
E_i = \sum_j E_{ij} = \sum_j |\vec{F}_{ij}|^2 = \sum_j (d_{ij} - r_{ij})^2,
\]  
(35)

and the error sum in whole network analog to total energy of the mass spring model is

\[
E = \sum_i E_i.
\]  
(36)

The positioning error of node \( i \) is reduced, when it is moved to the direction of resultant force (33).

By using mass-spring model analogue, it is possible to find the optimized localization estimates analog to minimum energy stage of the mass-spring model. There are two big challenges when applying this method in practice. First one is that how to get rough initial estimates to the node positions, and second one is how to avoid local minimums in optimization [18].

6. Other possible methods

6.1 Assisted satellite based positioning

As mentioned earlier, non line of sight problems, price and power consumption makes satellite based positioning systems difficult to use in ad-hoc sensor networks. Similar
problems makes satellite based positioning, currently GPS, also unsuitable in mobile phone localization, but in that area assisted GPS (A-GPS, sometimes typed just AGPS) gives new opportunities.

In A-GPS, the computation needed in GPS positioning is divided between cellular network and mobile phone. For example at the beginning of GPS position calculation, in so-called “cold start”, the one-cell accuracy of the mobile location is accurate enough to perform remarkable part of the computation needed at the beginning. Thus that part of computation can be done in the network, and then send the results to the mobile phone. Compared to pure GPS this leads to remarkable savings in the power consumption of the mobile phone, and also all of the equipments needed in GPS positioning system are not anymore needed in the mobile [19]. Qualcomm has even reported to get their A-GPS system operating in some lightly shadowed locations [20].

When the algorithms used in A-GPS develop further, it could be possible to reach so remarkable savings in power consumption and equipment costs that assisted satellite-based positioning becomes one possible alternative to localization also in some ad-hoc sensor network solutions.

6.2 Databases and digital maps

One can measure the fading of the radiosignal send from different known locations of the area, and store this information to the database. That database can be connected with digital geographical map of the area. When later measuring the radiosignal fading transmitted from some unknown location in the area, one can compare the fading value to the values stored in database, find the value that fits best and respective geographic location.

When talking outdoor environment, the dynamic nature of the radioenvironment make these kind of model unusable. The static fading model is not valid for long time, because changes in the weather etc. cause changes also in the radio environment. Furthermore, other complex effects in radioenvironment, like multipath, cause more problems.

In indoor positioning the situation is different, because there the radioenvironment is more stable. The size of the network is also much smaller than the size of outdoor cellular network, and hence more accurate mapmaking in the initialization phase makes sense. Digital maps and database localization method can be useful in some indoor ad-hoc network applications, and for example Finnish company Ekahau [21] is offering commercial localization product, where these ideas are exploited.

7. Example: Collaborative Multilateration

7.1 Initial configuration and phases

Collaborative multilateration, presented by Savvides et al [10], is an example of ad-hoc sensor network localization with beacons. At the beginning we have sensor network, where nodes are randomly scattered. Specified fraction of nodes called beacons are aware about their location, the rest called unknowns doesn’t have any knowledge about their locations.
Nodes have capability to perform ranging, and ultrasound is used in ranging measurements.

There are three main phases in collaborative multilateration: 1) formation of computational subtrees, 2) computation of initial estimates and 3) position refinement. The position estimates of unknowns that do not meet the criteria to belong in any of the computational subtrees are computed later in a post-processing phase [10].

7.2 Formation of Computational Subtrees

In the first phase, collaborative subtrees are configured. Collaborative subtree is a configuration of unknown and beacon nodes, where every unknown node has unique possible position estimate. Observe the situation first in one- and two-hop cases.

According to first condition to have a unique possible solution, it is necessary that an unknown node is connected to at least three nodes that have unique possible solutions. These nodes are not required to be beacons. Instead, unknowns needs to determine which of them neighbours have only one possible position solution, and use those nodes as reference points to determine weather they have unique position solution [10].

If the nodes used as a reference for unknown node lie in a straight line, the unknown will have two possible solutions and thus the location estimate is not unique. Based on this, we get the second condition: It is necessary for unknown node to use at least one reference node that is not collinear with the rest of its reference nodes. The positions of reference nodes are not known, but it is possible to estimate this condition by using cosinum theorem [10].

\[ |AC|^2 = |AB|^2 + |BC|^2 - 2|AB||BC|\cos(\angle ABC) \]

Figure 7. The use of cosinum theorem.

Assume that nodes A, C and D have unique position solutions in figure 7, and node B tries to figure out if its position solution is unique. B can measure its distances from nodes A, C and D. Distances AC and CD are also known. Using those distances B computes the angle ABC plus CBD, and then solves the distance AD by using that angle and cosinum theorem. If distance AD computed by using cosinum theorem is equal to measured sum AC + CD,
nodes A, C and D are collinear and hence node B doesn’t get unique position estimate by using those nodes.

One type of setup that can cause symmetry problems is shown in the figure below.

Nodes 3 and 4 both have three links to nodes with unique positions, but the setup is symmetric. Nodes 3 and 4 can be swapped with each other without any violation of the results reached from intra-node distance measurements. To avoid these type of situations, third condition is set: *In each pair of unknown nodes that use the link to each other as a constraint, it is necessary that each node has at least one link that connects to a different node from the nodes used as a references by the other node* [10].

Rules presented here can be extended to the case where unknown nodes are located $n$ hops from the beacons. Starting from the unknown node, it is tested weather it has at least three neighbours with tentatively unique positions. If the three neighbours do not already know if their solution is unique, then a recursive call is executed to each neighbour to determine if its position is unique. Each node used as an independent reference is marked used, that prevents other nodes re-use that node as an independent reference [10].

### 7.3 Computation of initial estimates

After formatting the collaborative subtrees in the first phase, the initial estimates of unknowns are computed in the second phase. Observe a two-hop case presented in figure 9.
Let the x-coordinate of beacon A be \(x_a\) and x-coordinate of beacon B \(x_b\). If the measured distance between unknown C and beacon A is \(a\), then the x-coordinates of C are bounded to the range \([x_a - a, x_a + a]\). Similarly, if the measured distance between unknown D and beacon B is \(b\), then the x-coordinates of C are bounded to the range \([x_b - (b + c), x_b + (b + c)]\). From these two ranges we choose the tightest left side limit and tightest right side limit, and get the bounds

\[
x_C \in [x_a - a, x_b + (b + c)]
\]

(37)

to the x-coordinate of unknown C. By using similar procedure, we get the limits to the y-coordinate of unknown C (and z-coordinate in three dimensions). After that the initial estimate of the node location is set in the middle of the bounding box. To obtain these bounding boxes, the locations of all the beacons are forwarded to all unknowns [10].

### 7.4 Position refinement

In the third phase of collaborative multilateration, the initial node position estimates computed in phase 2 are refined. This computation can be done in central node, or it can be fully distributed to every node [10].

By using the computational subtrees determined in phase one, and initial estimates determined in phase two, we get a well-determined of over-determined set of equations, which can be solved by using non-linear optimization.

For example, in the case presented in figure 10, we get following set of equations:

\[
\begin{align*}
f_{2,3} &= r_{2,3} - \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\
f_{1,3} &= r_{1,3} - \sqrt{(ex_3 - x_1)^2 + (ey_3 - y_1)^2} \\
f_{3,4} &= r_{3,4} - \sqrt{(ex_3 - ex_4)^2 + (ey_3 - ey_4)^2} \\
f_{4,5} &= r_{4,5} - \sqrt{(ex_4 - x_3)^2 + (ey_4 - y_3)^2} \\
f_{4,6} &= r_{4,6} - \sqrt{(ex_4 - x_6)^2 + (ey_4 - y_6)^2}.
\end{align*}
\]

(38)
In (38), $r_{i,j}$ is the measured distance between nodes $i$ and $j$, and $f_{i,j}$ is the error (residual) between measured distance and distance computed from position estimates. Notation $ex_i$ means the estimate of the coordinate $x_i$.

Objective is to minimize the mean square error over all equations of (38)

$$F(x_1, y_1, x_2, y_2) = \min \sum f_{i,j}^2.$$ (39)

There exists several standard methods how to solve the minimum least square problem (39). In the case of position refinement by computing in central node, this is done by using a Kalman Filter [10].

In Kalman Filter, we use the equations

$$x_k^- = A \hat{x}_{k-1} + Bu_k$$ (40)

$$P_k^- = AP_{k-1}A^T + Q$$ (41)

$$K = P_k^- H^T (HP_k^- H^T + R)^{-1}$$ (42)

$$\hat{x}_k = K_k (z_k - H\hat{x}_k^-)^{-1}$$ (43)

$$P_k = (I - K_k H)P_k^-$$ (44)

In the equations (40)-(44), $\hat{x}_k^-$ includes the initial estimates of locations. It is based on the known model of the system behaviour presented by matrix $A$. $u_k$ is a zero-mean gaussian random variable, $B$ is the error covariance matrix of that variable. $P_k^-$ is the a priori estimate of the error covariance and $Q$ is the process noise. $K$ is the Kalman Filter gain (Kalman gain), $z_k$ includes the measurements and

$$\hat{z}_k = H \hat{x}_k^-$$ (45)

is the predicted measurement. In (45), $H$ is the Jacobian of $\hat{z}_k$ [10].

During the collaborative multilateration, the network is assumed to be static. Thus the time update phase (40)-(41) in Kalman Filter is not used. To estimate the unknown locations, the algorithm proceeds as follows [10]:

i) Set the vector to initial estimates

ii) Evaluate the measurement update phase (42)-(44)

iii) Evaluate the stopping criteria
The stopping criteria is

\[ \sqrt{(\hat{x}_k^2 - (\hat{x}_k^-)^2)} \leq \Delta \]  

(46)

where \( \Delta \) is predefined tolerance.

iv) If the criteria (45) is met, algorithm determinates and gives the new position estimate. Otherwise set the prediction \( \hat{x}_k^- \) to new estimate \( \hat{x}_k \) and return to step ii.

For example, in the case described in figure 10 and equations (38)-(39), the initial estimates are

\[ \hat{x}_k^- = [ex_3, ey_3, ex_4, ey_4], \]  

(47)

and the ranging measurements are

\[ z_k = [r_{2,3}, r_{1,3}, r_{3,4}, r_{4,5}, r_{5,6}]. \]  

(48)

Vector \( \hat{z}_k \) contains the ranging distances based on current estimates:

\[ \hat{z}_k^T = \left[ \begin{array}{c} \sqrt{(x_1 - ex_3)^2 + (y_1 - ey_3)^2} \\ \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ \sqrt{(ex_4 - ex_3)^2 + (ey_4 - ey_3)^2} \\ \sqrt{(ex_4 - x_3)^2 + (ey_4 - y_3)^2} \\ \sqrt{(ex_4 - x_6)^2 + (ey_4 - y_6)^2} \end{array} \right] \]  

(49)

and H is the Jacobian matrix of \( \hat{z}_k \) with respect to \( \hat{x}_k^- \).

It is possible to achieve accurate node positions by using Kalman Filter in the position refinement, but that method is only suitable for centralized computation, because such computation cannot be performed by low-cost microcontroller available on the sensor nodes. Thus a distributed approximation of that method is developed for computing at every node.

In distributed scheme, the idea is similar to using a distributed Kalman Filter, but there are also some differences. Instead of using distributed Kalman Filter, the approximation in which nodes do not exchange covariance information is used. The computation is driven by ad-hoc networking protocols. After completing the first two phases, each node inside the computational subtree computes estimate to its location. In the case where unknown node is not directly connected to beacons, it uses the location estimates of its neighbours as reference points, and computes new estimate for it location. After that, unknown node broadcasts its new estimate to its neighbours, and its neighbours use it to update their own position estimates. This iteration is repeated until all the nodes in the network reach a pre-specified tolerance (46) [10].
If the iteration process is done uncontrolled, the nodes will converge to local minima and erroneous position estimates will exist. For avoiding this, the multilaterations at each node are executed in a sequence across all the unknown nodes in computational subtree. The sequences are repeated until all the estimates converge under the pre-specified tolerance (46). Thus, by using this distributed approach sensor nodes can together solve a global optimization problem that an individual node cannot solve [10].

8. Challenges

In many cases, the network area is assumed to be square- or circle form and isotropic. In the case when network area is not convex, the situation is more complicated. In those cases the robustness of many localization algorithms is not as good as in simplified cases.

Two remarkable error sources in localization are ranging measurement error and the effect of geometry. In some cases the selection of neighbours used in the sensor positioning has remarkable effect to positioning accuracy [5]. Thus some kind of algorithm how to make the selection so that the best possible positioning accuracy is achieved is needed.

Dynamic nature of sensor should keep on mind when developing localization algorithms. In the cases where sensors are moving, tracking and continuous update of position estimates must be available.

References


