Localization: Classification

Motivation

Standard GPS (Global Positioning System) receivers are not yet options for sensor nodes since they are much too expensive and large. For static nodes the positioning is also only needed in the beginning. But in many cases the usefulness of measurements is increased if their location is known. This makes positioning and localization essential however not trivial.

What is the difference between positioning and localization?

Positioning:	Determination of global world-coordinates
Localization:	Determination of relative coordinates

Localization: Classification

Classification

There are two dimensions: Nodes are position aware/unaware and nodes can/can not determine the distance to a neighbor.

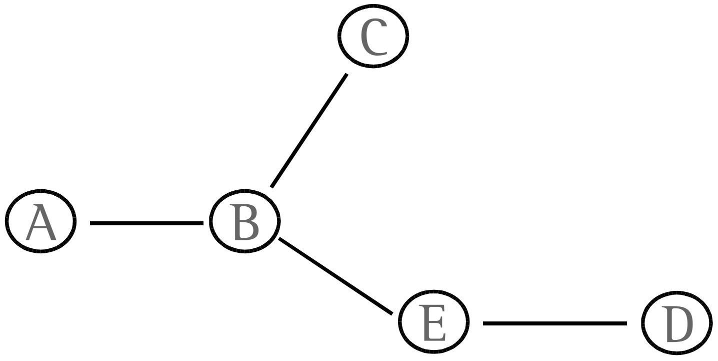
Distance to neighboring senders known / Some nodes know their position	yes	no
yes	complete topology in world coordinates	complete topology, but coordinate system only local
no	rough positioning within the neighborhood	only abstract connectivity matrix

Localization: Unknown positions and distances

No positioned node, no distance estimate

The single nodes don't know their global position. They can not estimate the distance of the neighbors within their radio range. Here only the creating of an adjacency matrix is possible.

	Node A	Node B	Node C	Node D	Node E
Node A	1	1			
Node B	1	1	1		1
Node C		1	1		
Node D				1	1
Node E		1		1	1



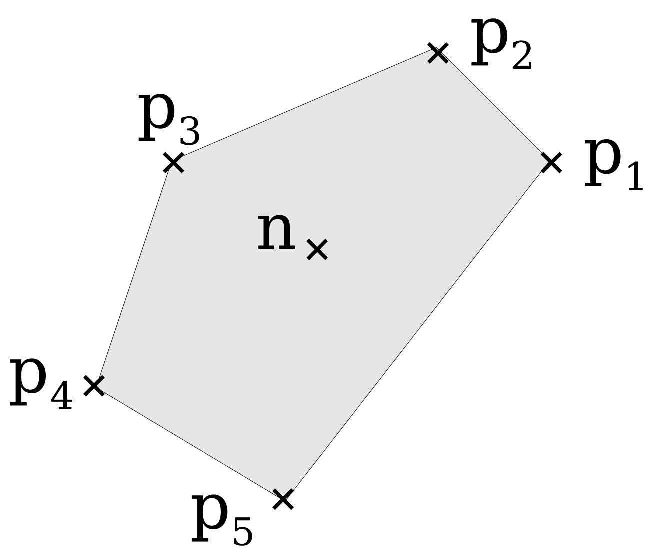
Localization: Known positions without distances

Simple global positioning

Create the average of the coordinates of the neighboring nodes within the range of receiver n.

Note: If the senders span a convex polygon, the median of the coordinates is the center of mass of the polygon.

$$n = \frac{1}{|P|} \sum_{\forall p_i \in P} p_i$$



Localization: Known positions and distances

Simple global positioning

Embedded systems in cars or cellular phones and PDAs could be nodes of a network, even though they may have been designed for other tasks. They often have better energy sources or are charged regularly.

Solution for static nodes: Sensor networks could use these embedded systems as location servers. Alternatively, among a huge number of cheap sensors, some expensive ones could be mixed supporting GPS. Then the simple senors could localize themselves using the complex ones. The GPS supporting nodes only need to be alive for a short period or must be available in the neighborhood at least for a while until the fixed nodes have determined their position.

Or some of the simple nodes could be exposed manually. Before being dropped a human operator has to obtain the exact coordinates e.g. using GPS. These manually positioned nodes can serve as location server for the rest of the network, at the expense of a greater initialization effort.

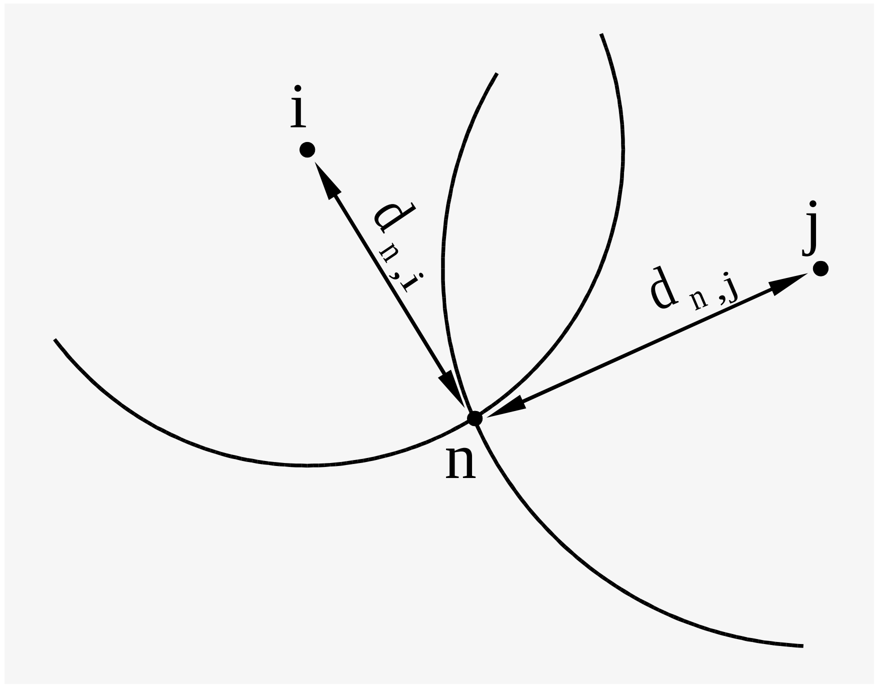
For moving nodes (e.g. attached to animals, in the sea etc.): as common in nautics, static public beacons could be used.

Localization: Known positions and distances

Simple global positioning

Basically three nodes are enough to calculate the position of a new node, provided that the mutual distances are known. The distances of the nodes among each other can be estimated e.g., via the strength of the incoming radio signal, which the receiving node detects.

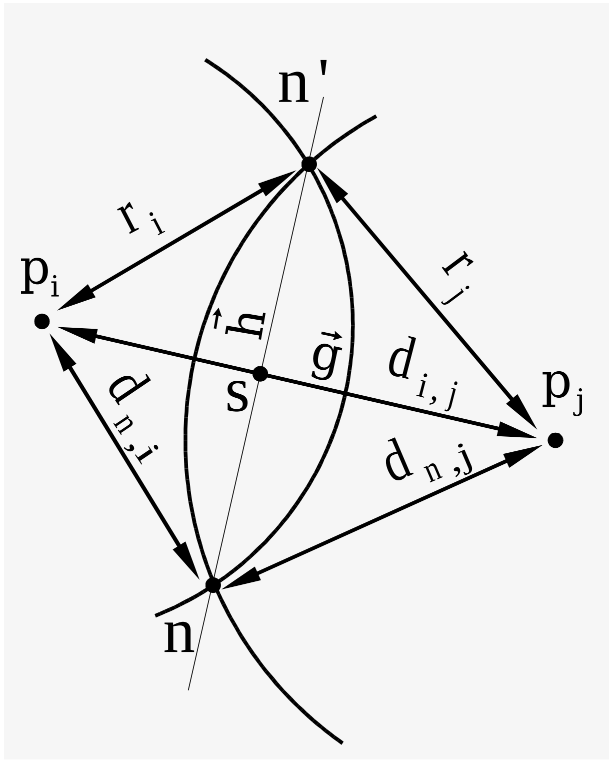
In 3D three nodes are sufficient for positioning if we can assume that the position must be on the surface of the sphere.

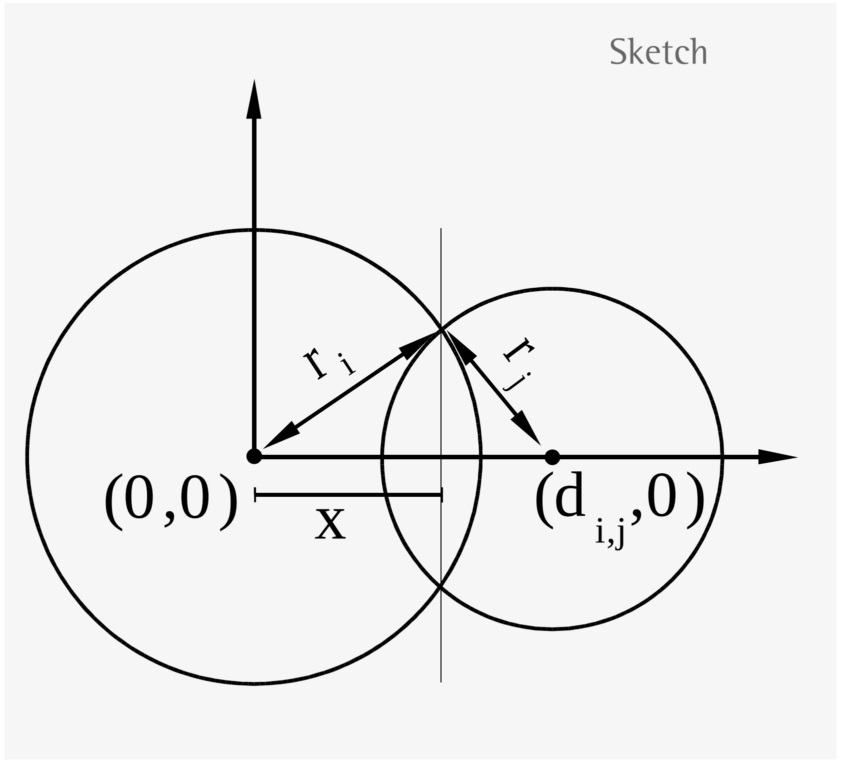


Localization: Known positions and distances

Simple global positioning

Two nodes i, j are given with the positions pi and pj respectively. The circle around pi through the searched node n corresponds to the distance di,n (analog for node j). In order to reach n, the positions s and and the distance ds,n as well as the perpendicular h to g are needed (di,j corresponds to the length of g). In order to be able to commit oneself to one of the two nodes, a third neighbor or other kinds of neighborhood information is needed.





Localization: Known positions and distances

Simple global positioning

Points on circle around (0,0):

$$x^2 + y^2 = r_i^2$$

Points on circle around (d,0):

$$(x - d_{i,j})^2 + y^2 = r_j^2$$

Calculation of the shared x-coordinate by insertion:

$$(x - d_{i,j})^2 + (r_i^2 - x^2) = r_j^2$$

$$x^2 - 2\,d_{i,j}\,x + d_{i,j}^2 + r_i^2 - x^2 = r_j^2$$

$$x = \frac{d_{i,j}^2 - r_j^2 + r_i^2}{2\,d_{i,j}}$$

Localization: Known positions and distances

Simple global positioning

p_i and p_j denote the coordinates of the nodes i and j. The point s in the lenticular intersection of the two circles results from:

$$s = p_i + x(p_j - p_i)$$

The perpendicular h to g is unique in 2D except for the yet undetermined sign:

$$\vec{h} = \pm \begin{pmatrix} -g.y \\ +g.x \end{pmatrix}$$ and the normalized lot to length 1 respectively $$\vec{h}^* = \frac{\pm 1}{|\vec{h}|}\vec{h}$$

Test:

$$\begin{pmatrix} g.x \\ g.y \end{pmatrix} \vec{h} = \begin{pmatrix} g.x \\ g.y \end{pmatrix} \pm \begin{pmatrix} -g.y \\ +g.x \end{pmatrix} = \pm(-g.x \times g.y + g.y \times g.x) = 0$$

Localization: Known positions and distances

Simple global positioning

We now obtain two candidates n and n’ from the product +/- d_{sn} h^* together with the point s . The distance d_{sn} between s and n results from the right triangle between p_i, s and n.

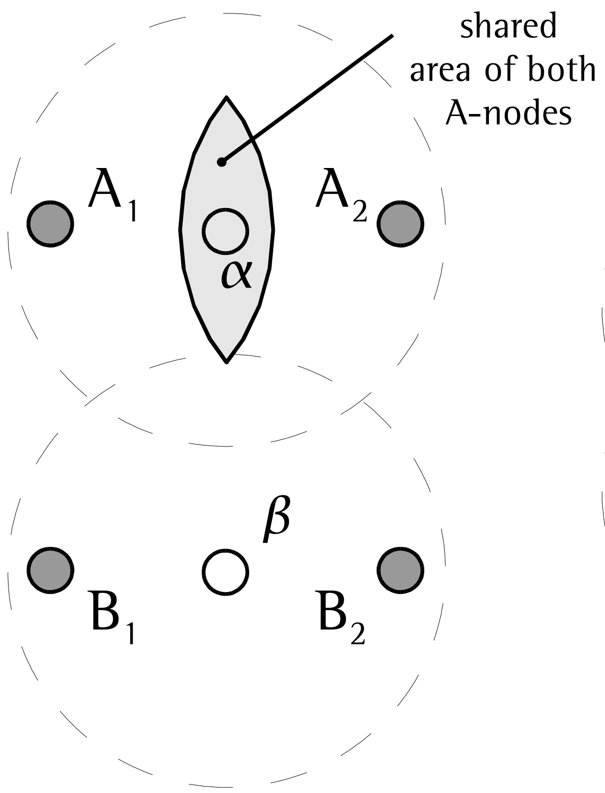
In the last step we have to find out, which one of the alternatives n or n’ is the true location. Usually this can be checked via a simple comparison with another neighboring node, only by checking the mutual “visibility”. Among the two alternatives for n, we are only interested in the one for which all neighbors are within the radio range.

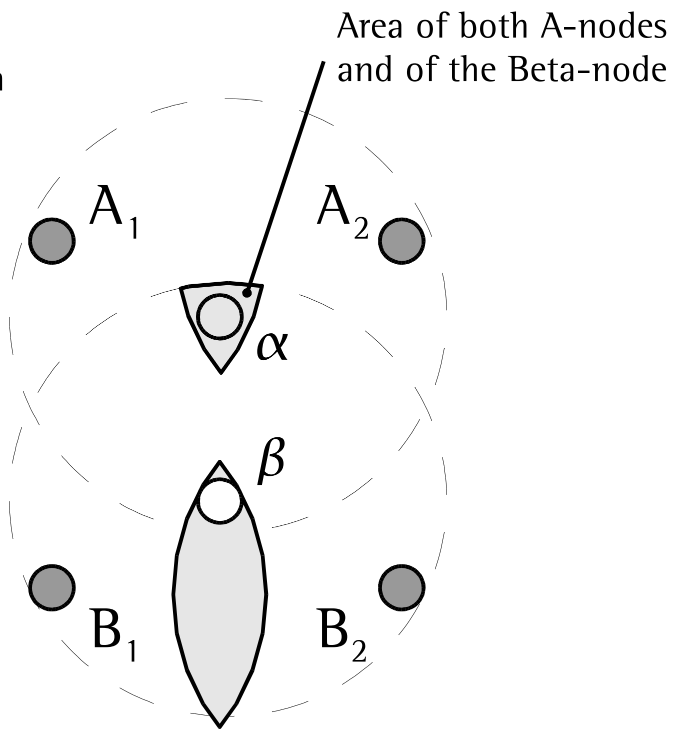
In 3D the positioning works analog, but with the help of three nodes with known coordinates. In the satellite based positioning the discrimination of cases is not necessary if it can be assumed that the position has to be on top of the earth (and not in space).

Localization: Known positions without distances

Some known positions with/without distance measurements

At a first view only the consideration of the neighbors with known coordinates is useful. At a closer look we see that the reachability of fuzzy positioned neighbors can shrink the area of uncertainty as well. If node beta is at the northern end of its valid area, it still cuts the area of node alfa, although the actual position of beta is unknown.





Localization: Known positions with distances

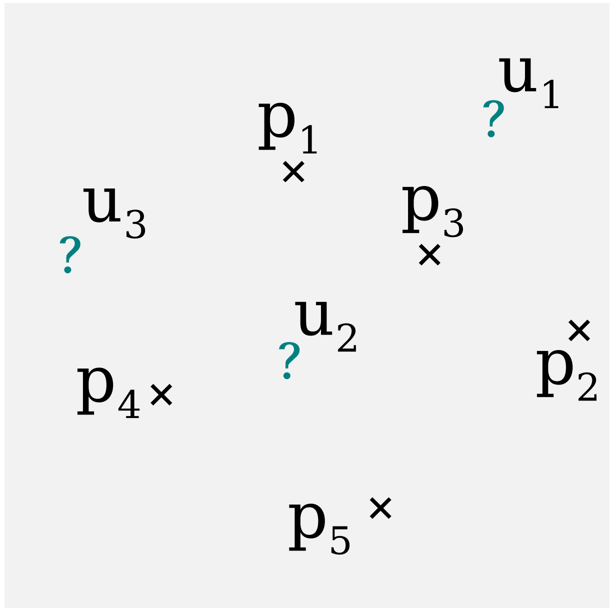
Some known positions with/without distance measurements

The problem of the positioning of nodes in a set of positioned and not yet positioned nodes can be expressed as optimization problem as follows.

$$\arg\min_{\forall(u_1, u_2, \ldots, u_k)} \sum_{\forall d_{i,k}} \left| \|u_i - x_k\|_2 - d_{i,k} \right|$$

u_i indicates the **unknown coordinate** of a node i, p_j the **determined coordinate** of a positioned node j. X indicates a node from the set union of both nodes.

$d_{i,j}$ denotes the distance between two nodes i and j, which can be derived e.g. from the field force of the received signal.



Localization: Known distances without positions

Motivation

The following approach is based on the work of Capkun, Hamdi and Hubaux

„GPS-free positioning in mobile ad-hoc networks“

It establishes a global coordinate system (CS) of the network based on mutual distance measurements. A central node is picked as origin of the CS.

Every node has to generate its own local coordinate system first, in which only its direct neighbors are included. What follows is the consolidation of the many local CS to a single global one. However, we can not determine how this CS is positioned in the real world and how it is rotated, because all calculations based on mutual distances are ”self-reliant” and no anchorage with world coordinates exists.

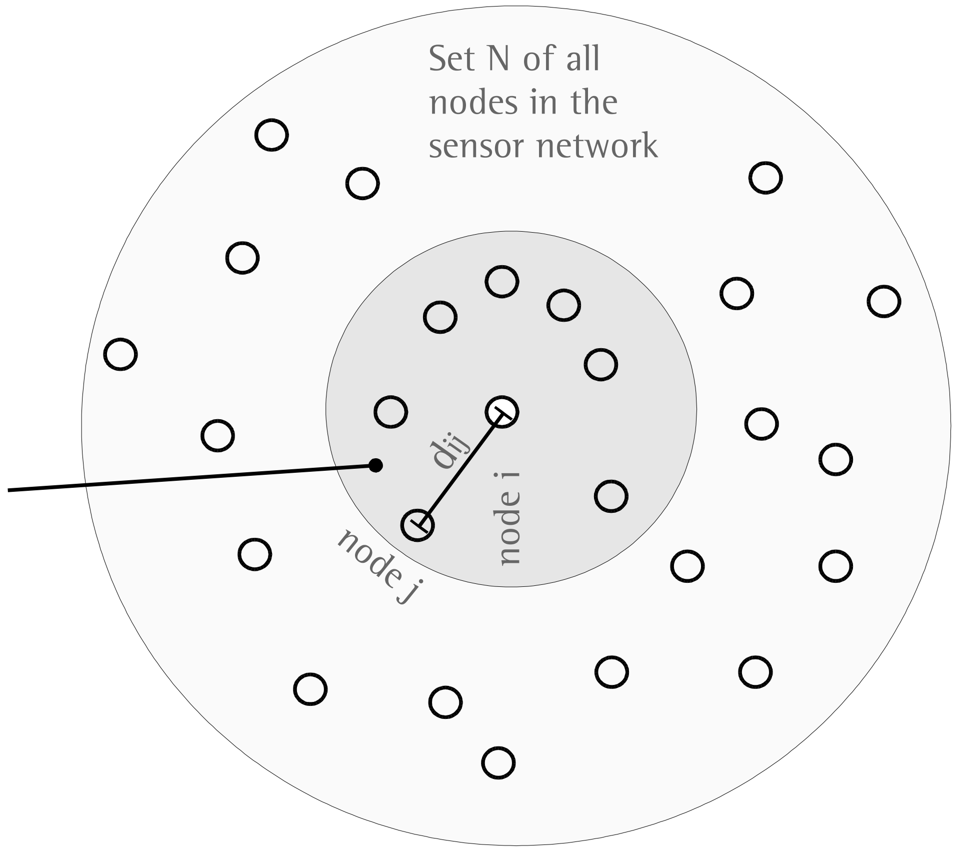
If however, the global (and relative) CS is detected, two globally localized nodes suffice to interpret all other nodes in world coordinates.

Localization: Known distances without positions

Requirement

Distances between the nodes can be derived from...

(a) the strength of the received signal
(b) the round-trip time of a beacon
(c) sending of acoustic signals



Local View Set (LVS): Set Ki of neighboring nodes of the node i (or 1-Hop neighborhood)

d_{ij} = distance between nodes i and j

Localization: Known distances without positions

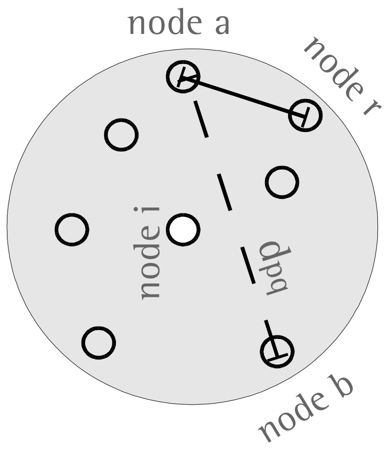
Determination of a local coordinate system (1)

(1) Every node determines its local neighbor, resulting in the 1-hop neighborhood. A direct radio contact exists between them.

(2) Every node sends the distance vector table from (1) to all its neighbors. Every node knows...

- its 1-hop and 2-hop neighborhood
- the distance of the direct neighbors
- some distances of its direct neighbors among each other, but not all of them!

Node “a” knows e.g. its neighbor “r”. It also informs node i about that. However, node a and b do not know each other. That’s why they can not inform node i about their distance. Consequence: i knows only some distances among its direct neighbors



Localization: Known distances without positions

Determination of a local coordinate system (2)

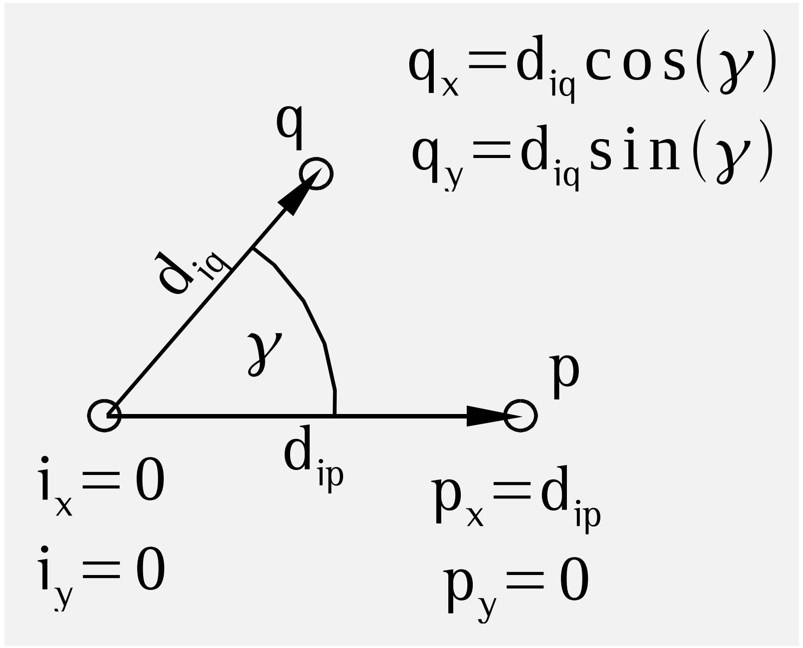
(3) In the beginning, every node determines a local coordinate system so that itself is in the origin.

(4) A node p from the LVSi is chosen, which defines the direction of the x-axis. Thereby the actual position of the node (in world coordinates) is irrelevant. So by definition the node p is now on the x-axis with the coordinates $(d_{ip},0)$.

Another node q is chosen, which helps defining the y-axis. As the y-axis is already defined as the perpendicular on the x-axis in 2D, q's task is only to determine the direction of the axis.

In order to be able to actually express q in coordinates, the angle gamma is missing.

Note: At the moment q does nothing more than implicitly defining the direction of the y-axis.



Localization: Known distances without positions

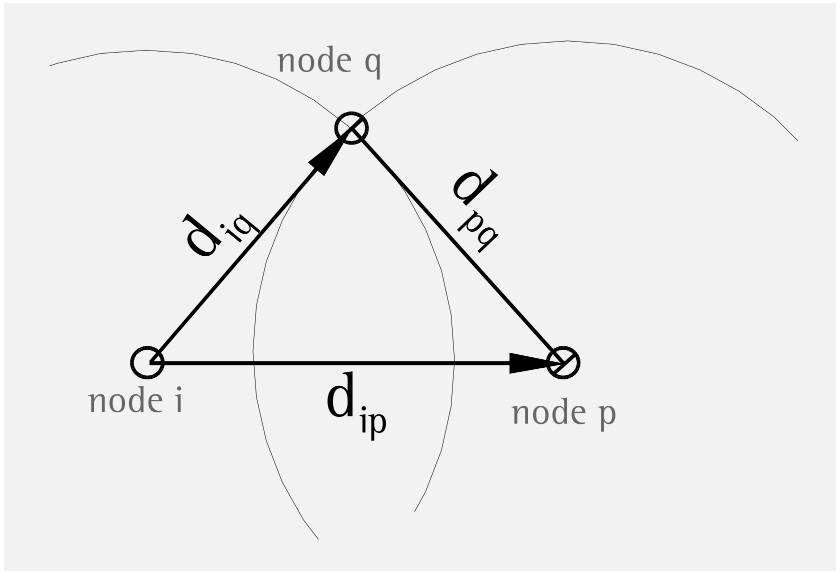
Determination of a local coordinate system (3)

Determine node q geometrically:

We get q by cutting the two circles

around node i with radius d_{iq} and
around node p with Radius d_{pq}.

For the calculation of the coordinates from node q the angle gamma (see previous page) and the distance d_{iq} suffice. This can be derived from the so-called cosine-rules.

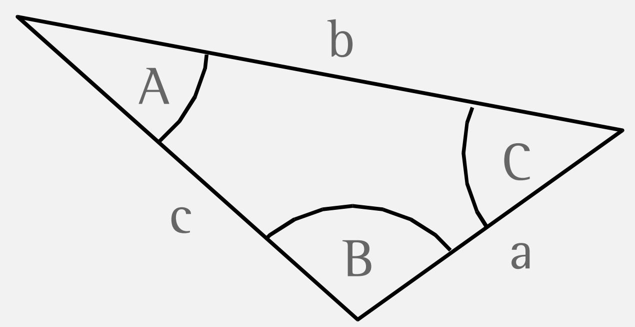


Localization: Geometric correlations

Cosine Rule - slide

In each version an angle and the length of the two involved sides of a triangle is needed. Hence the length of the side lying opposite to the vertex can be calculated.

$$a^2 = b^2 + c^2 - 2bc \times \cos(A)$$
$$b^2 = a^2 + c^2 - 2ac \times \cos(B)$$
$$c^2 = a^2 + b^2 - 2ab \times \cos(C)$$



Example:

$$a = 4{,}3 \quad c = 6 \quad B = 103° \quad b = ?$$

$$b^2 = 4{,}3^2 + 6^2 - 2 \times 4{,}3 \times 6 \times \cos(103)$$
$$b = \sqrt{4{,}3^2 + 6^2 - 2 \times 4{,}3 \times 6 \times \cos(103)}$$

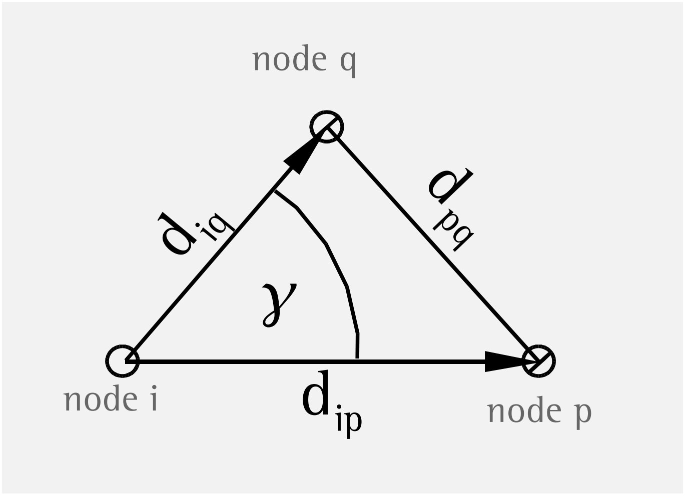
Localization: Known distances without positions

Determination of a local coordinate system (3)

$$d_{pq}^2 = d_{ip}^2 + d_{iq}^2 - 2\,d_{ip}\,d_{iq}\cos(\gamma)$$

$$2\,d_{ip}\,d_{iq}\cos(\gamma) = d_{ip}^2 + d_{iq}^2 - d_{pq}^2$$

$$\gamma = \arccos\left(\frac{d_{ip}^2 + d_{iq}^2 - d_{pq}^2}{2\,d_{ip}\,d_{iq}}\right)$$

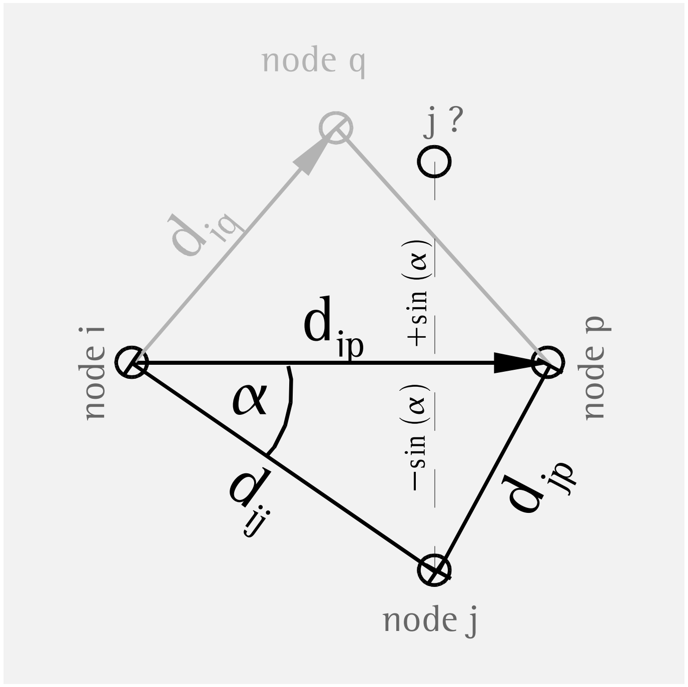


Localization: Known distances without positions

Determination of a local coordinate system (3): embedding of the visible nodes

In order to determine another node j, it seems we only need the nodes p and i. But mutual distances of the nodes provide no information about the side on which node j is located.

$$j_x = d_{ij}\cos(\alpha)$$
$$j_y = (\pm 1)\, d_{ij}\sin(\alpha)$$
$$\alpha = \arccos\left(\frac{d_{ij}^2 + d_{ip}^2 - d_{jp}^2}{2\, d_{ij}\, d_{ip}}\right)$$

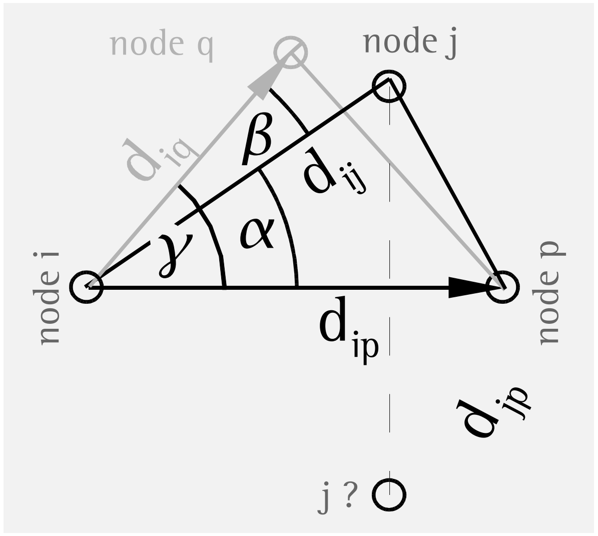


Localization: Known distances without positions

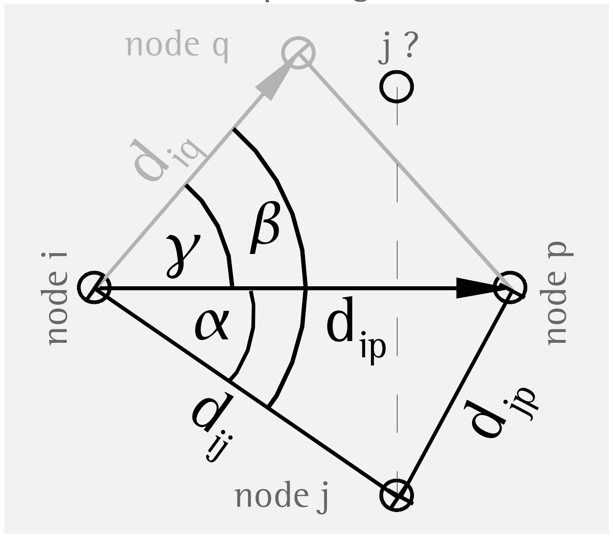
Determination of a local coordinate system (3): embedding of the visible nodes

Whether the node j is above or below the x-axis of the local coordinate system defined by i and p can be determined by analyzing the composition of angle beta:

Case: beta = gamma-alpha



Case: beta = alpha+gamma



$$\beta \approx |\alpha+\gamma| \Rightarrow j_y = -d_{ij}\sin(\alpha)$$
$$\beta \approx |\gamma-\alpha| \Rightarrow j_y = +d_{ij}\sin(\alpha)$$

Localization: Known distances without positions

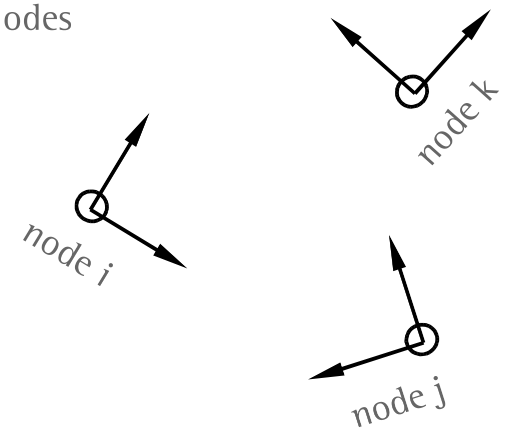
Consolidation of the coordinate systems (4)

A node k is always in the origin (0,0) of its own local coordinate system.
All nodes within its local view set (LVS_k) are interpreted in relation to k's position. K's world coordinates are (and remain) unknown.

However, k is also interpreted by the other nodes in the context of their coordinate systems. In the illustration at the bottom k is in i's LVS, thus it has a fixed position based on i's coordinate system. Furthermore with regard to j's coordinate system k has yet another position (though different to the position based on i).

Now we have to find a way to consolidate the local coordinate systems to a unique global one of the network. Here all nodes shall be interpreted with respect to node i. For that purpose the CS have to be rectified. They are rectified, if their axes point into the same direction.

Case 1: The rectification includes a rotation
Case 2: The rectification includes a mirroring
(often in addition to a rotation)



Localization: Known distances without positions

Consolidation of the coordinate systems (4): 1. case (adjusting of the direction)

All nodes in i's LVS are already contained in the CS of the network (with center i). We will now determine the coordinates of the 2-hop neighborhood which can not be seen directly from i (but e.g. from k). Therefor the other local CS have to be adjusted so that their children can also be included into the global CS.

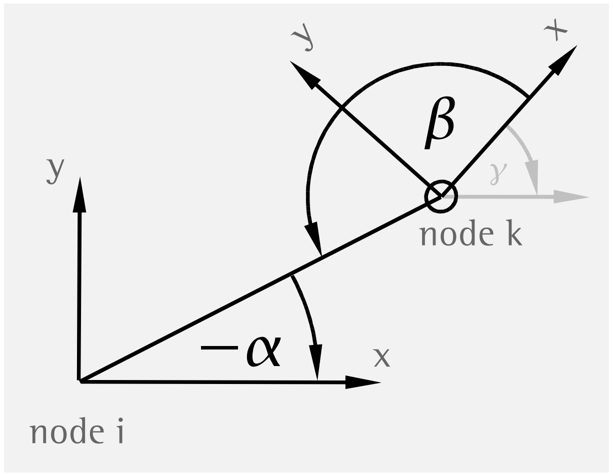
All coordinates seen by node k but not by i have to be rotated to fit into i's CS. Therefor the following operation has to be applied:

Given a coordinate q in the CS of k:

1) Rotate q by $-\beta+\alpha+\pi$ degrees
2) Add the coordinate of node k as seen by i.

$(\pi=180°)$

Note that in the sketch on the left a coordinate based on the view from node i is rotated back into the CS of k (any ideas for a better sketch?). What we actually want to do and what is described above is the opposite transformation from the view of k into the view of i.



Note: The proceeding on the left is based on the paper of Capkun et al. If node k knows either node i and the direction of i's x-axis it could also perform only one rotation (about the angle gamma encloded between k's x-axis and i's x-axis.

Localization: Known distances without positions

Consolidation of the coordinate systems (4): 2. case (mirroring of an axis)

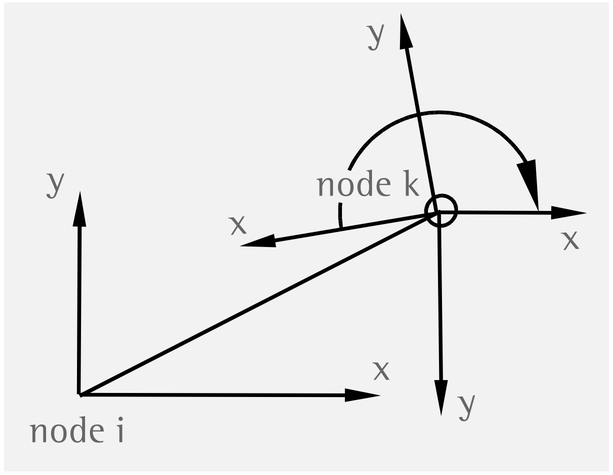
The single rotation seen on the previous slide does not suffice. Sometime the x-axis might be aligned with the other CS but the y-axis still points into the wrong direction.

After the rotation as shown on the right hand side the x-axis is aligned with the x-axis of node i however, the y-axis $\alpha+\beta$ still points into the wrong direction.

So in this case the y-axis has to be mirrored into the opposite direction. This means for the coordinates of the nodes in the local view set of node k that their y-components have to be multiplied by (-1).

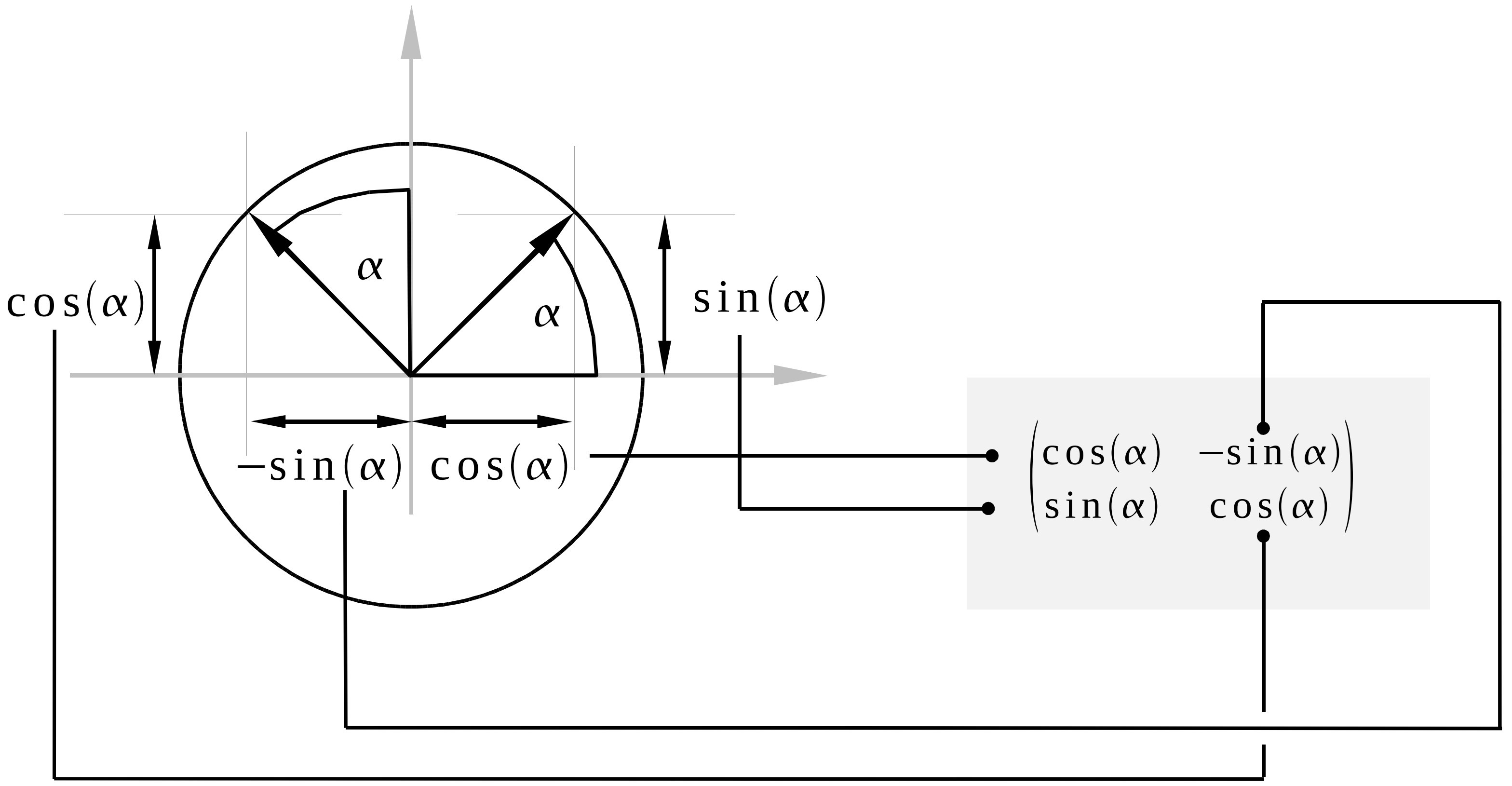
How do we get to know that this situation occured?

If the dot product of k's x-axis and i's x-axis yields to same sign as the dit product of the two knows y-axes than no mirroring is necessary.



Remember: projections and coordinate systems

Sine- Cosine- Rotation- slide



$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

Localization: Improved Localization

Improved global positioning

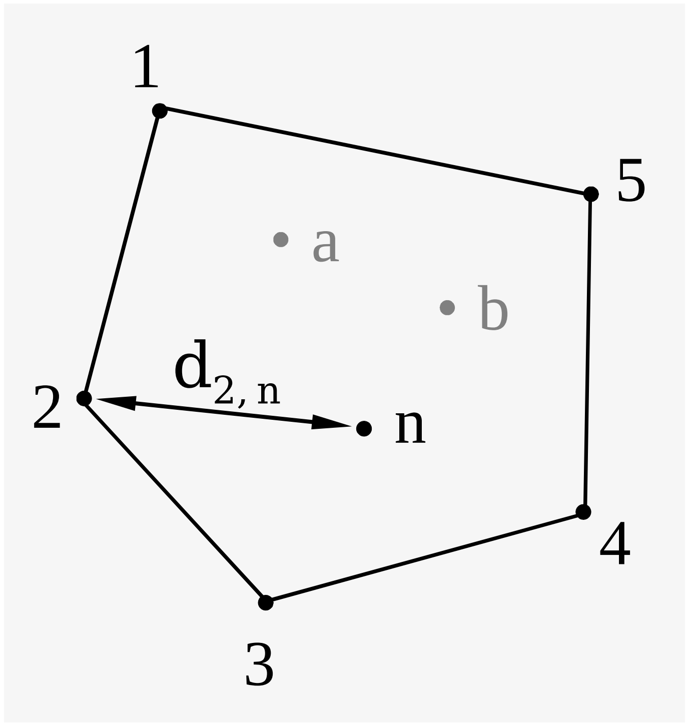
The position p_n of the node n is unknown. In n's LVS are several nodes (five in the Figure) with known positions. Now we want a) to include all nodes in the position calculation for n and b) within the position calculation to outweigh the influence of the nodes, which can estimate n's position best. As the field strength of a received signal degrades by the square of the distance, the distance to the closest nodes can be estimated best.

Given

n:	new node with unknown position
1-5:	nodes with known position (in general index k)
a, b:	more nodes, not on the convex hull They are ignored.
p_k:	(global) position of node k
$d_{k,n}$:	distance between node k and n. Estimated by the field strength of the incoming signal.

Searched

p_n:	position of node n



Localization: Improved Localization

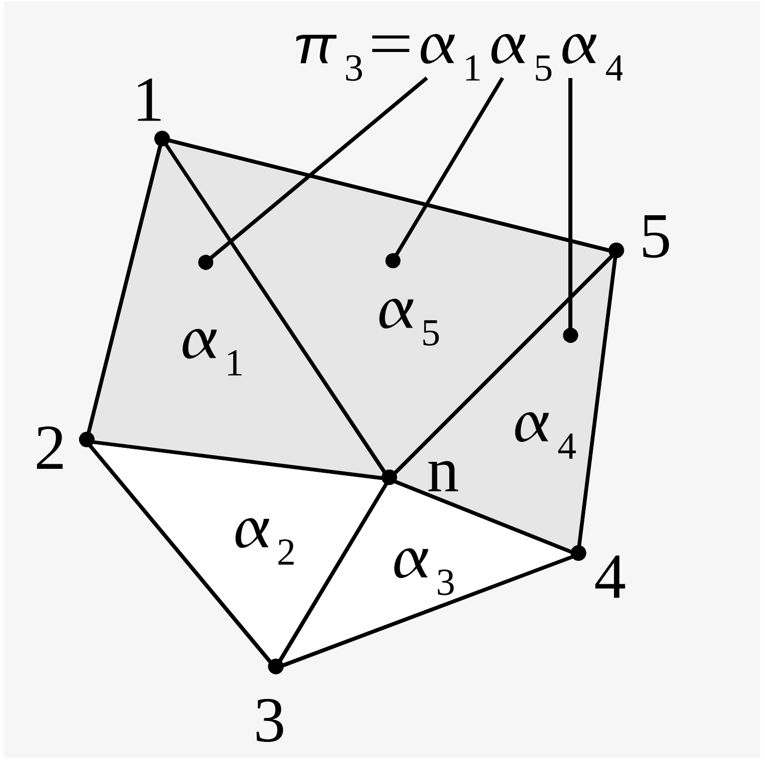
Improved global positioning

The position p_n shall be calculated as convex combination of the neighboring and yet localized nodes $p_1,\ldots, p_5$. Question: How do we have to choose the weights l_k so that a preferably optimal "mix" of the known positions is obtained which best approximates the real position of n?

Principle: If node n converges against e.g., the position of node 3 (see the Figure), then p_3 should have an increasing influence on the estimated position of n .

$$p_n = \sum_{k=1}^{K} l_k\, p_k$$

$$\sum_{k=1}^{K} l_k = 1$$



Localization: Improved Localization

Improved global positioning

The basis for the calculation of the weights l_k are the areas of the triangles, which are spanned by the yet unlocalized node n and respectively two neighboring localized nodes.

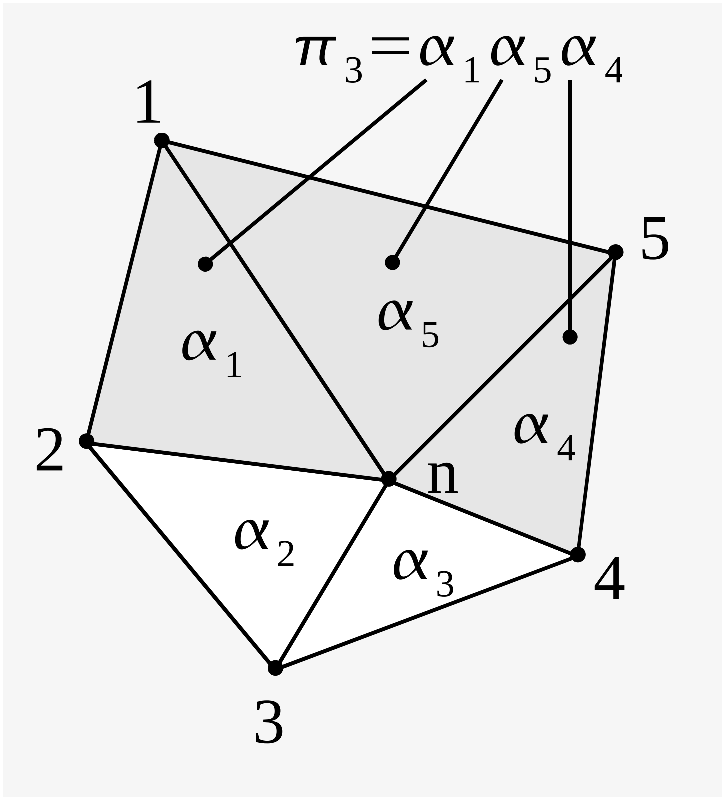
$$\alpha_k = \sqrt{(d_{k,k+1} + d_{k,n} + d_{k+1,n})(d_{k,n} + d_{k+1,n} - d_{k,k+1})(d_{k+1,n} + d_{k,k+1} - d_{k,n})(d_{k,k+1} + d_{k,n} - d_{k+1,n})}$$

The areas alpha_k can be calculated only using the side lengths of the triangles.

The actual weight of a node position p_k is composed of the product of almost every triangle.

$$\pi_k = \frac{\prod_{i=1}^{K} \alpha_i}{\alpha_{k-1}\alpha_k}$$

Only the two triangles, that border at the straight line between node n and k (nodes 3 on the right Figure) are excluded.



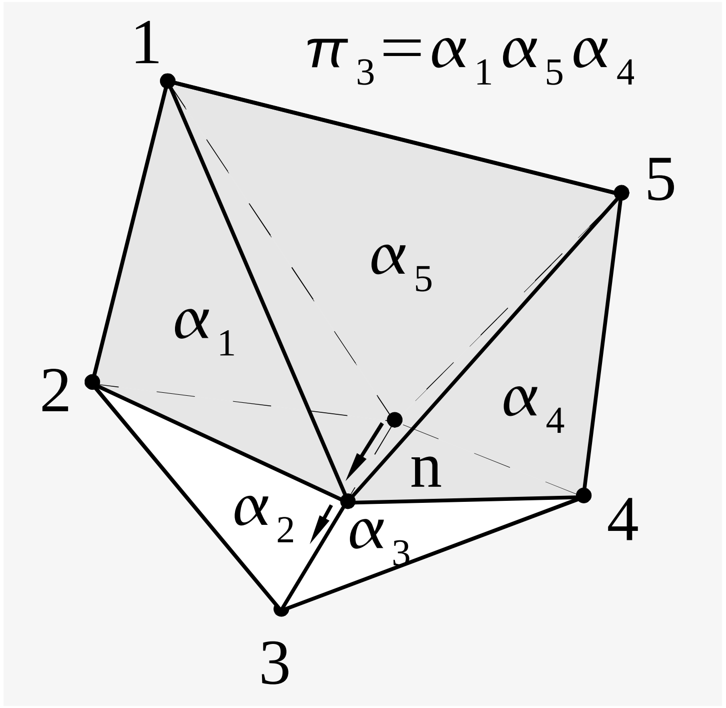
Localization: Improved Localization

Improved global positioning

Why have the two areas alpha$_2$ and alpha$_3$ no influence on the weight of node 3, although those areas are closest to node 3?

Answer: if n converges to node 3 in the example at the bottom, the weight pi$_3$ will converge against the weight of the complete polygon. As a result, the position p$_3$ has a growing influence on the calculation of the position p$_n$. The weights of the other nodes however, converge more and more against zero.
As node n approaches node 3, its position shall also dominate the estimate of p$_n$. At the same time the influence of the remaining nodes should vanish.

The pi$_n$ can not be used directly as weights. You have to normalize before, so that the sum of the weights is actually 1 (elsewise p$_n$ would not be within the polygon).



Localization: Improved Localization

Improved global positioning

The standardization of the weights pi to 1 in order to serve the convex combination is done here:

$$l_k = \frac{\pi_k}{\sum_{i=1}^{K} \pi_i}$$

In doing so the ratio of the weights among one another is not changed. Only the sum of the l-values is 1, the sum of the pi-values can be arbitrary. Now we want to determine the positions p_k weighted by the l-values. The result will be the yet unknown position of node n:

$$\xcancel{p_n = \sum_{k=1}^{K} l_k \, p_k}$$

For didactic reasons it has been omitted so fast that the simple convex combination above does not hold true (see the example program).

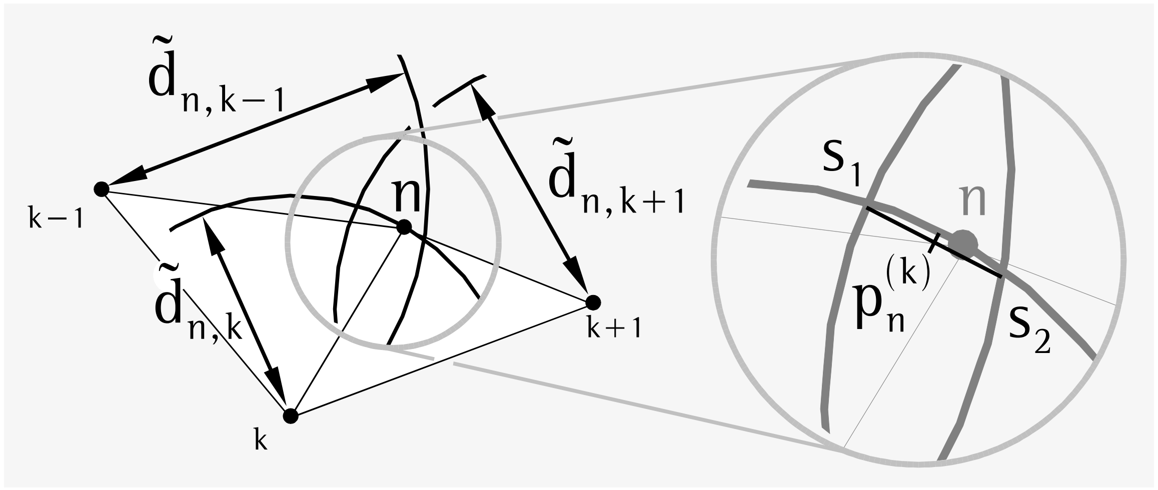
So instead of simply summing up the positions of the nodes p_k, each of these nodes has to estimate the real position of n. Then the final position does no more consist of the weighted sum of the node's positions themselves (as in the equation above), but of the estimates the nodes calculate.

Localization: Improved Localization

Improved global positioning

As shown at the beginning of the chapter two nodes are enough to calculate two candidates for the position of a third node. Being only interested in the candidates inside of the polygon, the candidate outside can be excluded easily.

Here node k shall estimate the position of node n. For that purpose it needs another node. Therefore it could choose its neighbor k-1 or k+1. It is even better to include both neighbors in the calculation of the position as depicted below. In doing so, k draws a circle around itself, which is touching nodes n (with radius $d_{n,k}$) (~ above d is a hint that d is only an estimate of the distance). Nodes (k-1) and (k+1) draw such a circle as well. If all estimations of distance d were exact, only one intersection would arise at the exact position of n. Exact distances can hardly be expected so that two intersections s_1 and s_2 evolve as shown below. The center of these two intersections is considered a suggestion from node k for the position of n and is weighted with the weight l_k. All other nodes perform the same calculation with their neighbors and weight their result with their weight l, so that in the end an optimized result for the position of n is obtained.


$\tilde{d}_{n,k-1}$
$\tilde{d}_{n,k+1}$
$\tilde{d}_{n,k}$
n
k − 1
k + 1
k
s_1
s_2
n
$p_n^{(k)}$