Corso di laurea Magistrale in Ingegneria dell'Automazione

RUUMBA: A RANGE-ONLY, UNSCENTED, UNDELAYED, MOBILE BEACON-ASSISTED FRAMEWORK FOR WSN DISCOVERY AND LOCALIZATION

RELATORE: Ch.mo Prof. Andrea Zanella

LAUREANDO: Davide Fauri

15-07-2013

A.A. 2012-2013

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Magistrale in Ingegneria dell'Automazione

RUUMBA: a Range-only, Unscented, Undelayed, Mobile Beacon-Assisted framework for WSN discovery and localization

Laureando:

Davide Fauri

Relatore:

Ch.mo Prof. Andrea Zanella

Anno accademico 2012/2013

Contents

Abstract

This work aims to analyze and implement some recent results on the SLAM problem, adapting them to the context of localizing a WSN using only RSSI distance measurements from a mobile beacon. The inherent system nonlinearities are dealt with by an Unscented Kalman Filter; newly discovered nodes are initialized cheaply and without delay by approximating their distribution with a Gaussian Mixture.

Furthermore, a series of static and dynamic path planning policies are developed and compared: their aim is to achieve a complete, precise, accurate network localization with minimum energy expenditure.

Chapter 1

Introduction

RUUMBA is a framework for WSN localization. But what is a **WSN**? The acronym stands for *Wireless Sensor Network*: it's a physical network, composed by a multitude of wirelessly connected *motes*: these small, cheap, autonomous hardware devices can perform limited computing and data storage, are often fitted with environmental sensing capabilities. From a topological perspective, a WSN can be compared to a graph (V, E) where the set of nodes V represents the motes, and each directed edge $e_{i,j} \in E$ is an established communication channel from mote i to mote j.

The fields of application are diverse: from natural disaster prevention, to mobile assets coordination, to indoor and outdoor area monitoring. In fact, due to their ease of installation, fault robustness and scalability WSNs are most efficient when deployed over rough or unaccessible terrain, or generally where any prolonged human intervention or supervision would be impractical.

Similarly, by attaching these nodes to a collection of generic objects it's possible to transfer these "smart" properties to it, thus considering the whole system as a *smart object network*, that can monitor its own condition or provide information about itself in a distributed manner. To provide some realistic, practical applications of this concept one can think of partially automated warehouses, where both human and *autonomous mobile robot* (AMR) workers operate simultaneously to frequently store, catalogue and retrieve smart goods; likewise, objects such as golf carts in an large airport or specific tools in a work area can be made smart too, for example by adding indicators for availability or maintenance.

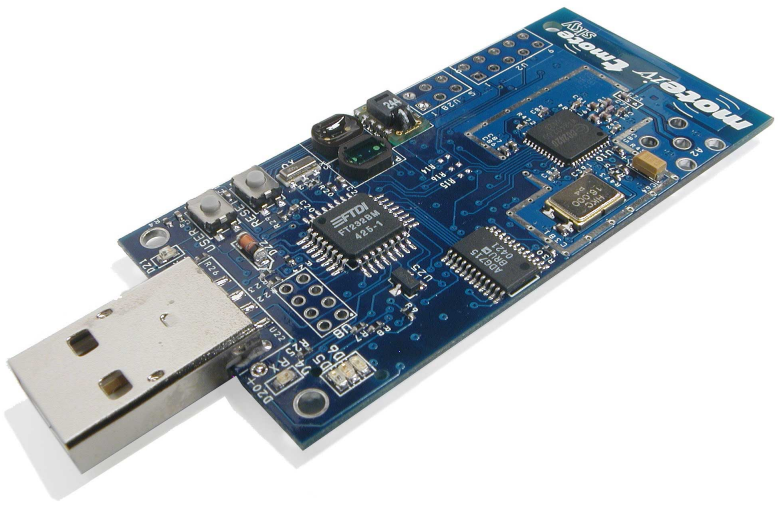


Figure 1.1: A typical mote of a WSN.

Common to all the above examples is a need for **localization**, i.e. acquiring knowledge about the spatial[1] coordinates of each node: as an additional difficulty, the nodes may possess a slow amount of mobility. Solving this requirement with an onerous initial calibration and installation, a human-supervised update, or by equipping each node with an expensive GPS device would obviously defeat the very same advantages offered by a WSN. These factors motivate the definition of a wholly automatic scheme or **framework** for discovering and localizing nodes, which can be executed repeatedly over time, with minimal additional hardware and with little to no prior knowledge about the network composition, topology and state.

A relatively recent approach pioneered by Sichitiu and Ramadurai [35] has been that of adding a mobile node to the network, aware of its position on a global coordinate system and capable of sensing and **assisting** with the localization of the other WSN nodes. This strategy is extremely cost effective, since a single

[1] In most cases, such as when the operative area spans the ground level or a single floor, planar coordinates are sufficient.

device can perform any amount of measurements and afterwards be redeployed over a different operative area; furthermore, scalability is achieved by limiting the information gathering to a local neighborhood. The data collected by this **Mobile Beacon** (MB) can then be filtered by an appropriate algorithm to obtain a position estimate for the whole network.

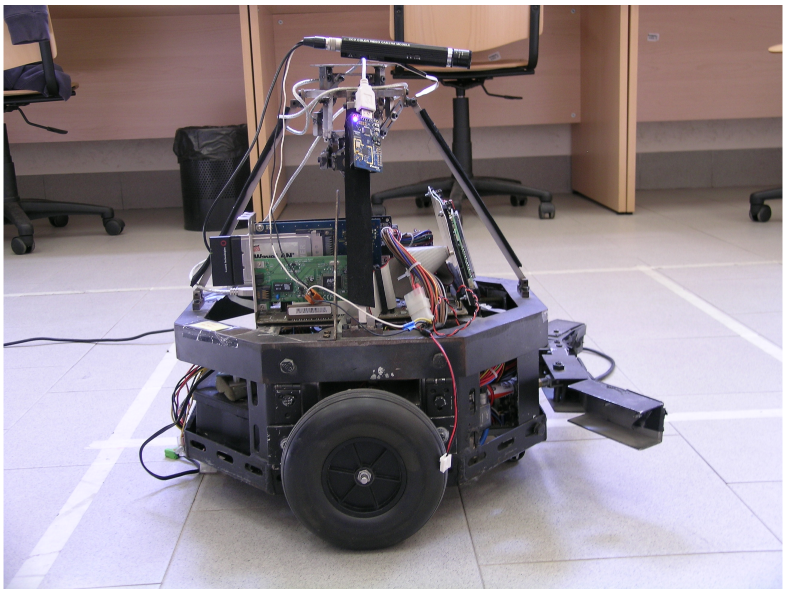


Figure 1.2: A typical example of a Mobile Beacon: a two-wheeled, differential drive robot equipped with a wireless antenna.

What data to gather, though? Since by definition each WSN node has wireless capabilities, it's possible to exploit this communication system to take *Received Signal Strength Indicator* (RSSI) measurements, which can be supported by all transceiver chipsets with minimal calibration. As signal strength is quite obviously correlated to the distance from the source, these **range-only** measurements represent a good compromise between informativeness and cost.

The algorithm choice is even more dependent on the specifics of each practical implementation: in a general case the nodes can be added, moved and removed during operations, so an online filter is needed. Moreover, their microcontrollers cannot perform difficult or large computations (sometimes, not even multiplication support can be taken for granted), so sophisticated procedures have to be executed by a base station with more processing and storage resources. By combining this base station and a MB into a single autonomous robot, the WSN localization problem is affine to the *Simultaneous Localization And Mapping* (SLAM) problem: the already efficient solution of adopting an *Extended Kalman Filter* (EKF) can be further improved by switching to the more recent **Unscented Kalman Filter** (UKF), better suited to dealing with the inherent nonlinearities of both motion and measurement models. The specific models that have been implemented for the scope of this thesis,[2] and the propagation and update filter equations are shown in Chapter 2.

During the process of exploring the whole operative area, the beacon may discover new nodes, which must then be added to the system state. This initialization must preserve the Gaussianity assumption made by the Kalman equations, while still describing accurately the spatial distribution implied by the measurements: standard solutions such as multilateration and particle filters introduce a delay in node initialization, and may be respectively inaccurate or computationally heavy. Modeling the initial distribution as a *Gaussian Mixture Model* (GMM) that will eventually converge, and evaluating all its components as independent hypotheses, allows for an **undelayed**, really easy to compute initialization which as an additional advantage can prove itself quite robust to noise in the initial measurements. To be integrated into this body of work, this technique had to be adapted to the specific case of Unscented RSSI localization: all the details are illustrated in Chapter 3.

Finally, the last step to optimize is the beacon points selection: to solve this experiment design problem, one should try to plan the path of the MB and the density of measurements along it in such a manner that the maximum amount of information can be extracted while expending the minimum amount of energy. The search for such an optimal path-planning **policy**, and the comparison of different alternatives both static and dynamic, constitutes the third and largest original part of this thesis and is treated in Chapter 4.

[2] Even though the UKF is agnostic to both of them.

All the simulations shown in Chapter 5 were coded using the *Matlab* language, due to its extensive library support and familiarity to the author.

For the rest of this thesis, a certain convention will be followed when graphically representing the localization process. The MB is depicted as a blue diamond moving on the planar map defined by the Cartesian axes; it leaves behind it a path of the same color, usually starting in the lower left corner. The ground truth of each WSN node position is drawn as a black asterisk: when a communication is established between the MB and a node and a corresponding RSSI range measurement is obtained, it's represented as a dashed magenta circumference having radius equal to the estimated distance. The whole state estimate $\hat{\mathbf{x}}$ is colored red: the mean positions of each node are triangles, while 95% confidence ellipses are drawn around them to represent their covariances.

In some figures, additional elements may be present: the robot's communication range, drawn as a dotted green line, represents the maximum distance between antennas at which a message can be expected to be received *assuming a uniform transmission channel*. It could happen that, due to shadowing effects discussed in section 2.3, a node outside this range can still be sensed by the MB.

As a visual aid for comparing localization errors, green error lines will associate the ground truth and the mean position estimate of each node; in a similarly unobtrusive manner, grid discretizations of the operative area will be drawn as a Voronoi map of yellow cells and center points.

Finally, when present, a black bounding box defines the invalicable limits of the node deployment area.

Chapter 2

Online range-only localization

USING only distance measurements to a node, find the node's position: to properly tackle this problem, it must first be reframed from an estimation point of view. Specifically, given a set $\mathbf{x}^b_{1,2,\ldots,M}$ of known beacon positions and a corresponding set $\mathbf{z}^n_{1,2,\ldots,M}$ of measurements about a node n, the aim is to reconstruct $\mathbf{x}^n$ as a vector containing the node's planar coordinates. Here, $\mathbf{z} = h(\mathbf{d}, \nu)$ is some statistic of the beacon-node Euclidean distance $\mathbf{d} = ||\mathbf{x}^b - \mathbf{x}^n||_2$, possibly disturbed by a noise factor ν whose Gaussianity and/or additiveness may not be guaranteed. This estimation should be conducted for each node of the network.

To analyze the statistics $\mathbf{z}$ with an online filter means to consider them one at a time, computing the update to the current state estimation each time a new measurement is received [32]; usually, this approach is reserved for dynamic systems whose underlying state changes over time and therefore must be constantly matched by the filter, or if the least number of measurements is required. In this latter case, the filter stops whenever a sufficient degree of confidence is reached.

In contraposition, offline localization methods such as multilateration, MultiDimensional Scaling [42] or Support Vector Machines [31] [49] all assume that every possible measurements is available at once, and perform a single computation taking all of them into account: even though these approaches cannot output partial results, therefore introducing an inevitable delay, they are usually more effective as noisy values can be averaged over all samples instead of a few.

The WSN localization problem is similar to SLAM: in both cases, a mobile robot creates an incremental map of its surroundings, storing mean estimate and covariance matrix of all the significant features in the environment, in our case the network nodes. These findings are used to both improve the belief about the robot's current internal state, and to plan the next step of its path. In our particular case, though, localization and mapping are treated as two separate processes: the AMR is already supposed to be able to localize itself without being conditioned on the extracted range information from the WSN, and the Maximum Likelihood estimate of its pose will be used as the MB position when localizing the nodes.

2.1 State

Without loss of generality, the time axis won't be assumed to be continuous but discrete; moreover, the length of the interval between a sample t and the successive one $t+1$ will not be constant. Time instead will be sampled nonuniformly after both motion and measurement phase have terminated, that is whenever the MB has moved from its previous position and all available measurements from the current position have been performed. One can look at t both as the number of stops made during the robot motion, and also as the incremental number of *virtual beacons* deployed on the operative area.

For a generic time instant t, the overall form of both system state $\mathbf{x}(t)$ and its estimate $\hat{\mathbf{x}}(t)$ is formalized as follows:

$$\begin{aligned} \mathbf{x}(t) &= \begin{bmatrix} \mathbf{x}_r(t) & \mathbf{x}_1(t) & \mathbf{x}_2(t) & \dots & \mathbf{x}_N(t) \end{bmatrix} \\ \hat{\mathbf{x}}(t) &= \begin{bmatrix} \hat{\mathbf{x}}_r(t) & \hat{\mathbf{x}}_1(t) & \hat{\mathbf{x}}_2(t) & \dots & \hat{\mathbf{x}}_{N'}(t) \end{bmatrix} \end{aligned} \tag{2.1}$$

where $\mathbf{x}_r(t)$, $\hat{\mathbf{x}}_r(t)$ represent the robot's actual and estimated state, and $\mathbf{x}_n(t)$, $\hat{\mathbf{x}}_n(t)$ are the actual and estimated planar positions of the n-th node. It's important to note the distinction between N and $N' \leq N$: the former number is the total number of WSN nodes to be localized, while the latter is the number of *discovered* nodes, that is the nodes from which at least a measurement have been received. For the rest of this chapter, it'll assumed that $N = N'$: the initialization of a new node into $\hat{\mathbf{x}}(t)$ will be dealt in Chapter 3.

Correspondingly, the error covariance matrix $\mathbf{P}(t)$ associated with $\hat{\mathbf{x}}(t)$ will be a $N' \times N'$ square block matrix with same dimension:

$$\mathbf{P}(t) = \begin{bmatrix} \mathbf{P}_r & \mathbf{P}_{r,1} & \mathbf{P}_{r,2} & \dots & \mathbf{P}_{r,N'} \\ \mathbf{P}_{1,r} & \mathbf{P}_1 & \mathbf{P}_{1,2} & \dots & \mathbf{P}_{1,N'} \\ \mathbf{P}_{2,r} & \mathbf{P}_{2,1} & \mathbf{P}_2 & \dots & \mathbf{P}_{2,N'} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N',r} & \mathbf{P}_{N',1} & \mathbf{P}_{N',2} & \dots & \mathbf{P}_{N'} \end{bmatrix} (t) \tag{2.2}$$

In fact, it's possible to achieve a more simplified form of (2.2) by considering the conditional independence of any two nodes' position estimates $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ with respect to the *inter-node distance* measure $\mathbf{z}_{i,j} = h(\mathbf{d}_{i,j}, \nu)$. The Bayes Nets shown in Figg. 2.1 and 2.2 illustrate this concept (light blue nodes are known or observed variables, arrows express dependence, while the path of the Bayes Ball algorithm [34], which implies correlation, is colored in red): when the only measurements available are those between the MB and some node, the corresponding node position estimate is unrelated from all other estimates. Contrariwise, acquiring knowledge of $\mathbf{z}_{i,j}$ brings information about $\mathbf{d}_{i,j}$, which being a sufficient statistic of the joint distribution of $\mathbf{x}_i$ and $\mathbf{x}_j$ necessitates the introduction of the terms $\mathbf{P}_{i,j}$ and $\mathbf{P}_{j,i}$ in the expression of $\mathbf{P}$.

By choosing to disregard these measures, it's possible to obtain a sparse block variant of (2.2) where no covariance is present between any two nodes:

$$\mathbf{P}(t) = \begin{bmatrix} \mathbf{P}_r & \mathbf{P}_{r,1} & \mathbf{P}_{r,2} & \dots & \mathbf{P}_{r,N'} \\ \mathbf{P}_{1,r} & \mathbf{P}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{P}_{2,r} & \mathbf{0} & \mathbf{P}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{N',r} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{P}_{N'} \end{bmatrix}(t) \tag{2.3}$$

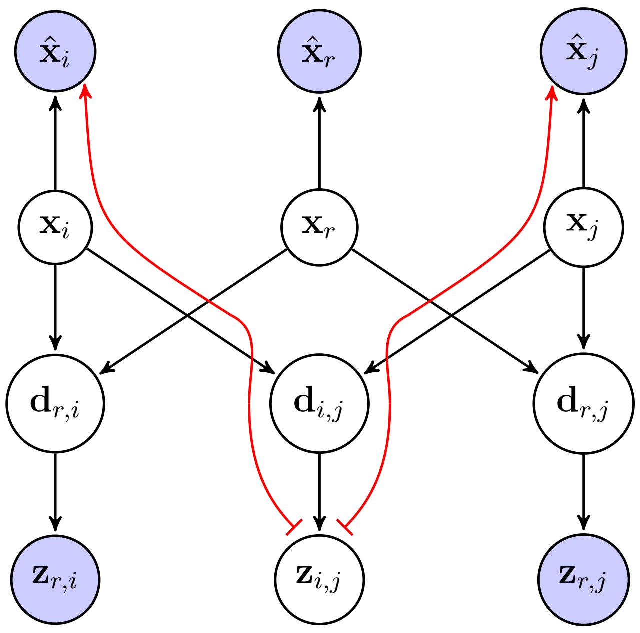


Figure 2.1: Path of the Bayes Ball algorithm where inter-node distance is not measured: there is no dependence between the two nodes' estimated positions.

2.2 Motion model

The robot state $\mathbf{x}_r$ is a vector of 3 elements, expressing its pose as a combination of coordinates on the Cartesian plane and an angle which represents the robot's current heading direction:

$$\mathbf{x}_r(t) = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}(t), \qquad \theta_r \in (0, 2\pi] \tag{2.4}$$

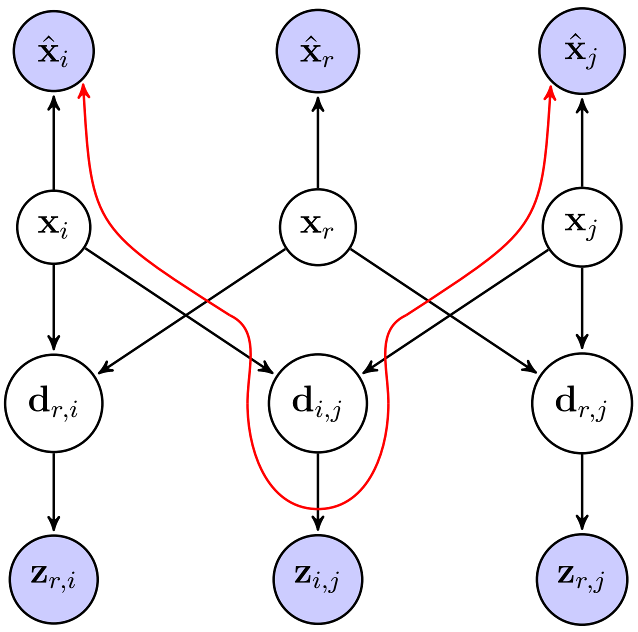


Figure 2.2: Path of the Bayes Ball algorithm where inter-node distance is measured: $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ are now correlated.

We consider our AMR to be a two-wheeled, differential drive robot (see Fig. 1.2), capable of in-place changes of direction without the need for an additional steering motion. Thus, every polygonal path can be decomposed into a series of consecutive rotations and translations, and the odometric inputs at each time step t can be written as $\mathbf{u}(t) = [\Delta\theta, \Delta D\,](t)$; these values are assumed to be disturbed by additive Gaussian noises ν_θ, ν_D with standard deviations σ_θ,σ_D.

The motion, then, is equivalent to a shift in polar coordinates centered on the robot position. The (nonlinear) update equations are:

$$\mathbf{x}_r(t+1) = \mathbf{x}_r(t) + \begin{bmatrix} (\Delta D(t) + \nu_D) \cdot \cos(\,\theta_r(t) + \Delta\theta(t) + \nu_\theta) \\ (\Delta D(t) + \nu_D) \cdot \sin(\,\theta_r(t) + \Delta\theta(t) + \nu_\theta) \\ \Delta\theta(t) + \nu_\theta \end{bmatrix} \tag{2.5}$$

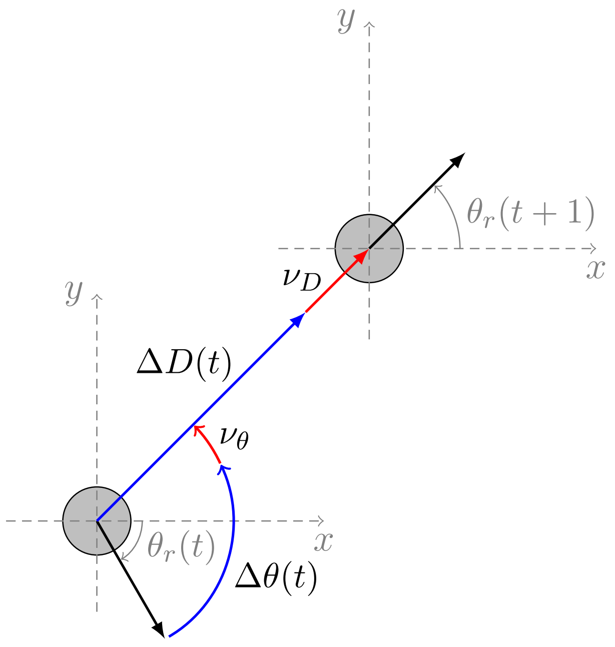


Figure 2.3: Motion model for a differential drive robot.

2.3 Measurement model

When a wireless radio signal is transmitted, the power measured at the receiver station decays exponentially with respect to the distance between the two antennas. To represent the received signal strength, almost all IEEE 802.11 enabled devices utilize the Received Signal Strength Indicator (RSSI), a generic power metric which uses arbitrary units (typically defined on a 0-100 interval). More often than not, there are no industry standards that precisely relate RSSI values to mW or dBm power levels; also, each vendor provides its own accuracy, granularity, and range specifications, creating the need for either a WSN composed only by the same kind of devices, or a shared precalibration phase for heterogeneous networks.

Having measured the RSSI over k messages, and linearly converted them into power samples P_{rx}^1, P_{rx}^2, $\ldots$, P_{rx}^k, one still needs to know the model of the transmission channel in order to estimate the distance between two devices. A common, general radio propagation model for RSSI ranging is the log-distance path loss model [9], which relates the power measured at the receiver station to the logarithm of the actual distance from the signal source:

$$\mathbf{P_{rx}}[\text{dBm}] = P_{tx}[\text{dBm}] - PL(d_0)[\text{dBm}] - 10\eta \log_{10}\left(\frac{d_{true}}{d_0}\right) + \mathbf{\Psi} \tag{2.6}$$

In the above expression, $\mathbf{P_{rx}}$ is the received power represented as a random variable, P_{tx} is the nominal transmission power of the emitting antenna, d_{true} is the Euclidean distance between communication endpoints and d_0 is the Fraunhofer minimum distance for an antenna to be in far field conditions (usually d_0 is between $0.1 - 1$ m). The remaining three terms are used to model the environmental properties of the channel: $PL(d_0)$ is the path loss measured at d_0, assuming a free space propagation; η is called the path loss coefficient, and its value is 2 when the signal travels in a vacuum, but spans ranges of $1.8 - 2.4$ for indoor propagation and $1.5 - 5$ for outdoor propagation; finally, $\mathbf{\Psi} \sim \mathcal{N}(0, \sigma_\Psi^2)$ is a Gaussian noise factor.

The model (2.6) is an average, computed over the whole area that can be reached by the emitter: local effects can greatly affect how the channel influences the measure. The most common interferences are due to multipath and shadowing phenomena: in the former case, the same signal travels not only on a straight line to the receiver, but is also being reflected by obstacles, walls, the ground, etc.: since the radio wave propagates along paths of different lengths, a signal can cause destructive or constructive interference with itself. In the case of shadowing, also known as slow fading, more local obstructions than the average do interpose themselves between the source and the receiver: the signal is then attenuated more strongly than predicted by the model; the inverse effect can happen when there are less local obstructions than the whole area average.

It's been proved experimentally [2] that by employing a multichannel transmission (for example, the 16 channels specified by the IEEE 802.15.4 protocol, spanning an interval of $2410 - 2480$ MHz), one can average the different RSSI values over the frequency space and significantly reduce the effect of multipath interferences. In fact, the overall noise deviation σ_Ψ, which in indoor environment is often equal or greater than $4 - 4.8$ dB, can be limited to a more conservative

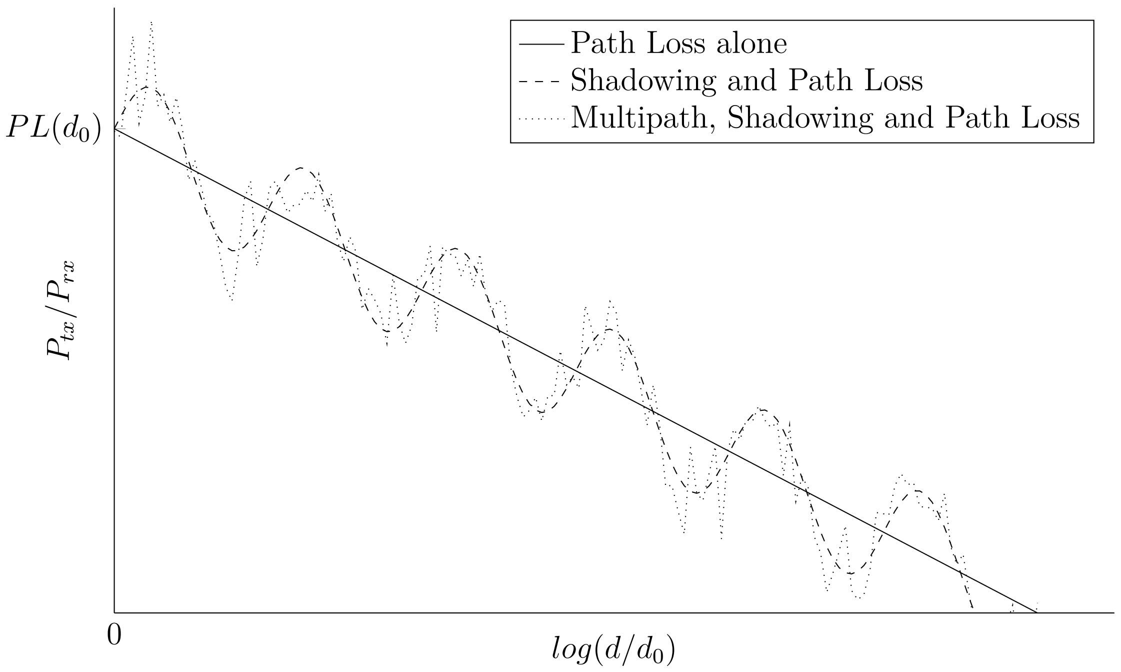


Figure 2.4: Graphical representation of the influence of environment on RSSI localization.

2.8 dB, with a gain of approximately 2 dB. Alas, medium-scale effects such as shadowing cannot be so easily dealt with, so $\mathbf{\Psi}$ remains a spatially correlated random variable: to actually describe it as statistical noise, i.e. to validate the assumption that its samples $\Psi_1, \Psi_2, ...\Psi_k$ are mutually independent, it's necessary to move the MB by a certain minimum distance between different measurements . Even though this depends on the scale of the operative area over which the WSN operates, the distance d_0 is usually a sensible choice.

By computing the mean received power $\bar{\mathbf{P}}_{\mathbf{rx}}$ via RSSI frequency average, it's possible to invert the equation (2.6) to derive the formula for the estimated distance distribution:

$$10\eta \log_{10}\left(\frac{d_{true}}{d_0}\right) = P_{tx} - P_{rx} - PL(d_0) + \mathbf{\Psi}$$

$$\hat{\mathbf{d}} = d_0 \cdot 10^{(P_{tx} - \hat{\mathbf{P}}_{\mathbf{rx}} - PL(d_0))/10\eta} \tag{2.7}$$

$$\hat{\mathbf{d}} = d_{true} \cdot 10^{\mathbf{\Psi}/10\eta} \tag{2.8}$$

While expression (2.7) may be more relevant for practical implementations, (2.8) is a more readable alternative for simulations; moreover, it can also be modified to show that $\hat{\mathbf{d}}$ is lognormally distributed:

$$\begin{aligned} \hat{\mathbf{d}} &= e^{\ln d_{true}} \cdot e^{\ln 10 \cdot \mathbf{\Psi}/10\eta} \\ &\sim \ln\mathcal{N}\left(\ln d_{true},\ \left(\frac{\ln 10}{10\eta}\sigma_\Psi\right)^2 \right) \\ &= \ln\mathcal{N}\left(\mu,\ \sigma^2\right) \end{aligned} \tag{2.9}$$

The mean of this estimator, though, is not unbiased. Even assuming to measure RSSI samples on evenly spaced points along a circular trajectory with radius of d_{true} around the emitting antenna, the resulting expected value is [45]:

$$\mathrm{E}[\,\hat{\mathbf{d}}\,] = d_{true} \cdot e^{\sigma^2/2} \tag{2.10}$$

This may cause problems in a realistic implementation: for example, even assuming constant values of σ^2 over all measures, the actual position of the yet unlocalized WSN node is not guaranteed to lie within the complex hull defined by the MB path. Since all distance estimates are longer than the ground truth, the final node position estimate will consistently differ by a proportionate amount.

Instead, a median estimator is preferred: using as plug-in values the Maximum Likelihood estimates of the lognormal location and scale parameters, namely $\hat{\mu} = P_{tx} - PL(d_0) - \mathrm{E}[\mathbf{P_{rx}}]$ and $\hat{\sigma}^2 = \mathrm{Var}[\mathbf{P_{rx}}]$, a recent paper by N. Longford [26] explicated the formula for an efficient unbiased median estimator whose general form is $\hat{\mathbf{d}} = \exp(\hat{\mu} + b\hat{\sigma}^2)$. In the paper, two variants were compared: the *UnbiasedN* estimator, that has been specifically computed as to have a bias of exactly 0, and the *UnbiasedA* estimator, which simply consists in the Taylor approximation of the previous one.

$$\hat{\mathbf{d}}_{uN} = \exp\left(\hat{\mu} + \frac{k-1}{2\hat{\sigma}^2}\left(1 - e^{\frac{\hat{\sigma}^2}{k(k-1)}}\right)\hat{\sigma}^2\right) \tag{2.11}$$

$$\hat{\mathbf{d}}_{uA} = \exp\left(\hat{\mu} - \frac{1}{2k}\hat{\sigma}^2\right) \tag{2.12}$$

As shown in Fig. **??**, there are no appreciable differences between the performances of the two estimators, both in terms of relative bias and in terms of variance: therefore, for ease of computation, the simplest one was chosen to be implemented. The formula for its Mean Squared Error (MSE) can also be found in [26], and is equal to:

$$\mathrm{Var}[\hat{\mathbf{d}}_{uA}] = e^{2\mu}\left(\frac{e^{2\hat{\sigma}^2/k}}{\left(1+\frac{2\hat{\sigma}^2}{k(k-1)}\right)^{\frac{(k-1)}{2}}} - \frac{e^{\hat{\sigma}^2/k}}{\left(1+\frac{\hat{\sigma}^2}{k(k-1)}\right)^{(k-1)}}\right) \tag{2.13}$$

However, as will be shown in Chapter 3, a more readable and tractable form of (2.13) would be preferred. Fortunately, the above equation can be rewritten at different degrees of simplification by the consecutive application of a binomial approximation and two Taylor approximations:

$$\mathrm{Var}[\hat{\mathbf{d}}_{uA}] \underset{bin.}{\approx} d_{true}^2 \cdot \frac{e^{\hat{\sigma}^2/k}}{1+\hat{\sigma}^2/k}\left(e^{\hat{\sigma}^2/k} - 1\right) \tag{2.14}$$

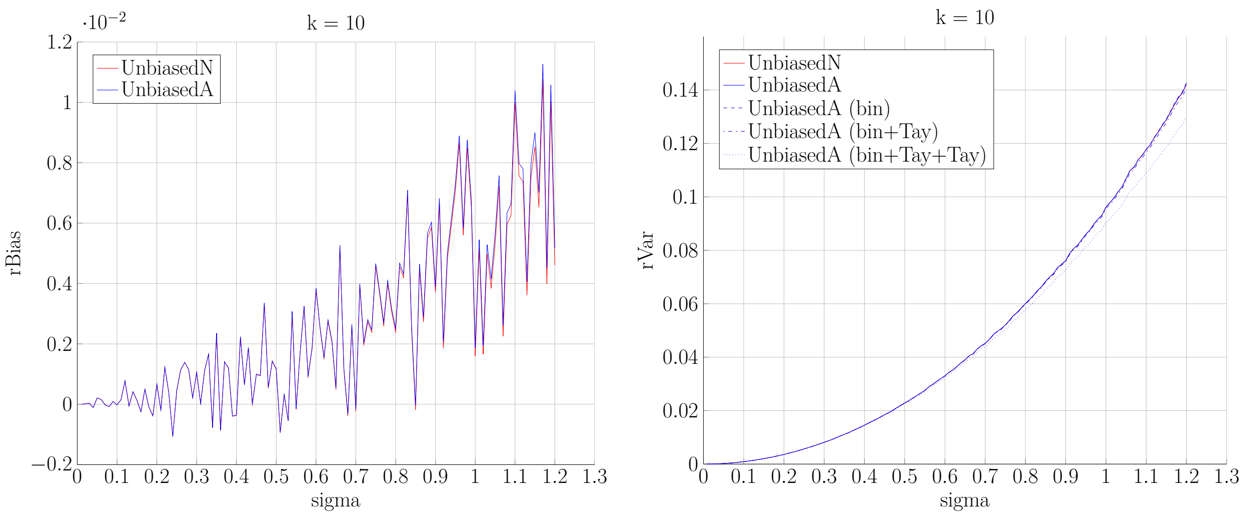
$$\underset{Tay.}{\approx} d_{true}^2 \cdot \left(e^{\hat{\sigma}^2/k} - 1\right) \tag{2.15}$$

$$\underset{Tay.}{\approx} d_{true}^2 \cdot \frac{\hat{\sigma}^2}{k} \tag{2.16}$$

A simulational analysis, averaged over 20000 experiments with $k = 10$ samples each, has been conducted to compare the goodness of formulae (2.11 - 2.16): the relative bias $rBias = \mathrm{E}[\hat{\mathbf{d}}]/d_{true} - 1$ and the relative variance $rVar = \mathrm{Var}[\hat{\mathbf{d}}]/d_{true}^2$ have been plotted against a realistic range of the scale parameter σ.

Even if (2.11) and (2.12) do not seem to differ from one another, they do both exhibit a slight bias due to the small number of samples (see Fig. 2.5a). Still, it's well below 1 percent when$\sigma < 1.1$, that is when $\sigma_\Psi < 9.5$ dB assuming a path loss factor of $\eta = 2$.

Regarding $rVar$, from Fig. 2.5b it's apparent that the only approximation that introduces a sensible deviation from the original variance is the second application of Taylor's formula: therefore, (2.15) offers the best compromise between tractability and exactness.



(a) Relative bias

(b) Relative variance

Figure 2.5: Comparison of lognormal median estimators, for different values of σ.

Finally, the likelihood of each new distance measure $\tilde{d}_n(t)$, taken at time t with respect to the n-th node, shall be computed too. As it will be used (see Chapter 3) by the online algorithm when updating the confidence weights associated with each position estimate $\hat{\mathbf{x}}_n$, it should express the probability of measuring the corresponding power value $\tilde{P}_{rx,n}(t)$ versus the expected distance to the same node, $\hat{d}_n(t) = \mathrm{E}[||\hat{\mathbf{x}}_r(t) - \hat{\mathbf{x}}_n||_2]$. Dropping the time notation for ease of comprehension, the equations are:

$$\begin{aligned} \ell(\tilde{d}_n \,|\, \hat{d}_n) &= \Pr[\,\hat{\mathbf{d}} = \tilde{d}_n \,|\, \hat{d}_n\,] \\ &= \Pr[\,\mathbf{P_{rx}} = \tilde{P}_{rx,n} \,|\, \hat{d}_n\,] \\ &= \Pr[\,\mathbf{\Psi} = 10\eta \log_{10}\left(\frac{\hat{d}_n}{d_0}\right) - (P_{tx} - \tilde{P}_{rx,n}) + PL(d_0) \,|\, \hat{d}_n\,] \end{aligned} \tag{2.17}$$

2.4 Extended Kalman Filter SLAM

In the commonly known version of the Kalman Filter, all states $\mathbf{x}$ are univariate or multivariate Gaussian distributions which are propagated and observed through linear functions described by the matrices F, G, and H, and by the inputs $\mathbf{u}$ and measurements $\mathbf{z}$. Those variables are disturbed by additive Gaussian noises $\mathbf{w}$ and $\mathbf{v}$, with zero mean and known covariances $\mathbf{Q}$ and $\mathbf{R}$:

$$\mathbf{x}_t = F\mathbf{x}_{t-1} + G(\mathbf{u}_{t-1} + \mathbf{w}_{t-1}) \tag{2.18}$$

$$\mathbf{z}_t = H\mathbf{x}_t + \mathbf{v}_t \tag{2.19}$$

Still, in most real cases the motion and measurement functions are nonlinear, so the Kalman Filter cannot be directly applied to predict and update the state estimate; moreover, noise additivity may not be guaranteed. A more general formulation, then, would be:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{w}_{t-1}) \tag{2.20}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{v}_t) \tag{2.21}$$

If f and h are differentiable, it's possible to linearize them around the current best state estimate $\hat{\mathbf{x}}_t$ by calculating their Jacobian matrices:

$$\mathbf{F}_{t-1} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t-1},\mathbf{u}_{t-1},\mathbf{0}} \qquad \mathbf{G}_{t-1} = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\hat{\mathbf{x}}_{t-1},\mathbf{u}_{t-1},\mathbf{0}} \qquad \mathbf{L}_{t-1} = \left.\frac{\partial f}{\partial \mathbf{w}}\right|_{\hat{\mathbf{x}}_{t-1},\mathbf{u}_{t-1},\mathbf{0}} \tag{2.22}$$

$$\mathbf{H}_t = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_t,\mathbf{0}} \qquad \mathbf{M}_t = \left.\frac{\partial h}{\partial \mathbf{v}}\right|_{\hat{\mathbf{x}}_t,\mathbf{0}} \tag{2.23}$$

Translating the familiar set of Kalman equations to this linearized system, the *first-order Extended Kalman Filter* (EKF) is obtained:

Predict

$$\hat{\mathbf{x}}_{t|t-1} = f(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_{t-1}, \mathbf{0}) \qquad \text{Predicted state estimate} \tag{2.24}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_{t-1}\mathbf{P}_{t-1|t-1}\mathbf{F}_{t-1}^T + \mathbf{L}_{t-1}\mathbf{Q}\mathbf{L}_{t-1}^T \qquad \text{Predicted covariance estimate} \tag{2.25}$$

Update

$$\mathbf{i}_t = \mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1}, \mathbf{0}) \qquad \text{Innovation} \qquad (2.26)$$

$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^T + \mathbf{M}_t \mathbf{R} \mathbf{M}_t^T \qquad \text{Innovation covariance} \qquad (2.27)$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^T \mathbf{S}_t^{-1} \qquad \text{Kalman Gain} \qquad (2.28)$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \mathbf{i}_t \qquad \text{Updated state estimate} \qquad (2.29)$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_t) \mathbf{P}_{t|t-1} \qquad \text{Updated covariance estimate} \qquad (2.30)$$

Since the state at time t consists of $2N' + 3$ elements (cfr. eq. (2.1) and (2.4)), the computational complexity of EKF SLAM when predicting is $O(2N')$, as only the robot state variance $\mathbf{P}_r$ and the robot-node covariances $\mathbf{P}_{r,n} = \mathbf{P}_{n,r}^T$ have to be modified for all nodes $n = 1, 2, \ldots, N'$.

However, the update step is dominated by the $O(4N'^2)$ covariance update of equation (2.30): repeating this computation over a map of size N drives the total cost of the algorithm to $O(N^3)$

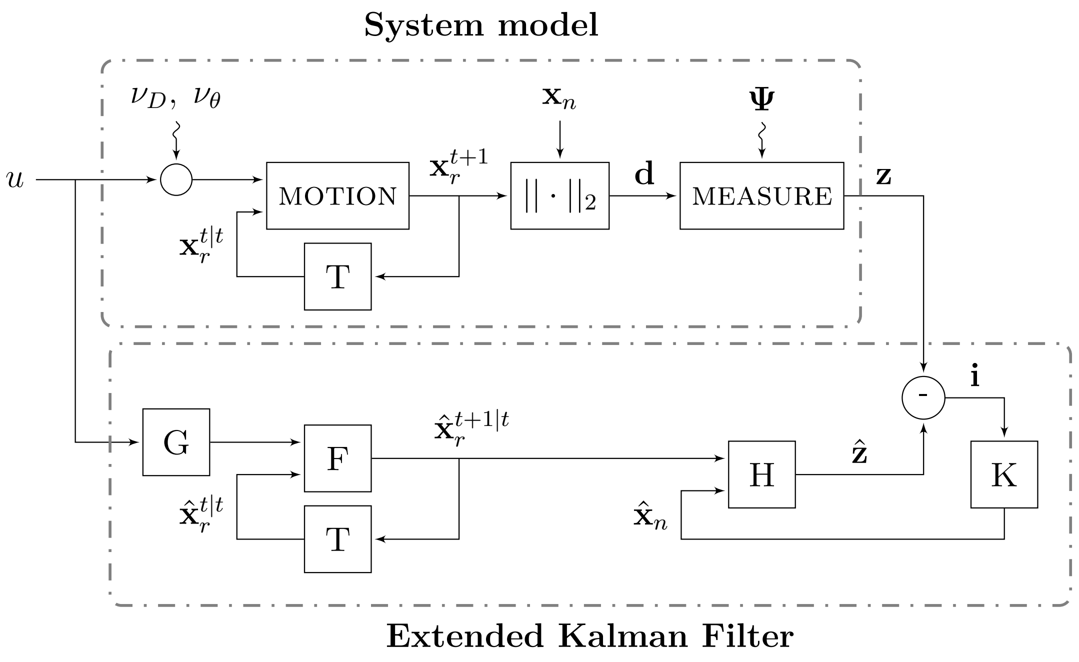


Figure 2.6: Block model for a typical EKF-SLAM.

2.5 Unscented Kalman Filter SLAM

Even if EKF-SLAM keeps being a widely used technique in the context of localization, it's still a first-order approximation of a nonlinear system,[1] and as such it's subject to large errors when comparing the true posterior to the state estimate. These errors tend to underestimate the covariance $\mathbf{P}$ and may lead to sub-optimal performance and sometimes filter divergence [43].

Monte Carlo alternatives (i.e. particle filters) would certainly obviate the problem, but to balance their inherent randomness a large number of particles would be required; by considering that each discovered node would have its own set of particles, this line of reasoning quickly becomes unfeasible.

The *Unscented Kalman Filter* (UKF) belongs to the family of Linear Regression Kalman Filters [23], and therefore can be thought of as performing an implicit statistical linearization of the motion and measurement models. Instead of sampling a lot of random points extracted from the underlying distribution, it propagates a small set of deterministically selected *sigma points* that can express all the posterior moments up to the 3rd order.

It should be noted that the state in a UKF still follows the Kalman assumption of Gaussianity, and hence the bivariate Gaussian which represents the planar position estimate of a generic node $\mathbf{x}_n$ is still an approximation. Yet, as shown by the comparison between 95% confidence ellipses in Fig. 2.7, the covariance is correctly estimated; this is due to the direct application of the formulae (2.20 - 2.21) instead of their Jacobians (2.22 - 2.23).

[1] While second-order versions of the EKF exist, their increased implementation and computational complexity tend to prohibit their use.

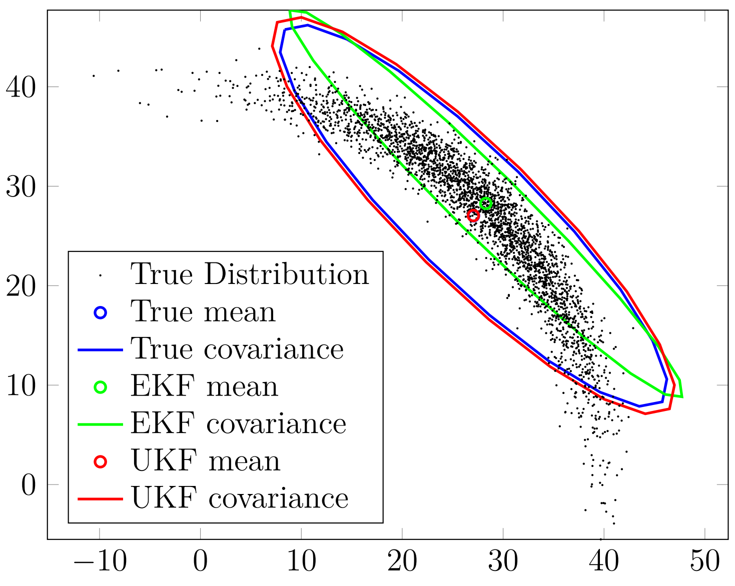


Figure 2.7: Comparison between particle filter, first-order EKF and UKF when noisily converting from a polar to a cartesian coordinate system.

2.5.1 State augmentation

For what concerns the motion and measurement noises, the original UKF formulation assumed additiveness: as this is not the current case, the original state estimate and covariance should be augmented to include the Gaussian noise factors due to odometry and shadowing:

$$\hat{\mathbf{x}}_a = \begin{bmatrix} \hat{\mathbf{x}} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{P}_a = \begin{bmatrix} \mathbf{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_\theta^2 & 0 & 0 \\ \mathbf{0} & 0 & \sigma_D^2 & 0 \\ \mathbf{0} & 0 & 0 & \sigma_\Psi^2 \end{bmatrix} \tag{2.31}$$

In this way, the propagation function will depend only on the total state $\mathbf{x}_a$; also, as it will be seen in the following paragraphs, by initially augmenting the state there is no need for a second derivation of the sigma points when advancing from the predict phase to the update phase [46].

2.5.2 Sigma points

At the heart of the UKF is the *Unscented Transformation*: this operation can capture the statistical properties of a generic random variable of size m by sampling $\Theta(m)$ vectors from it, distributed around the mean according to some function of its covariance matrix.

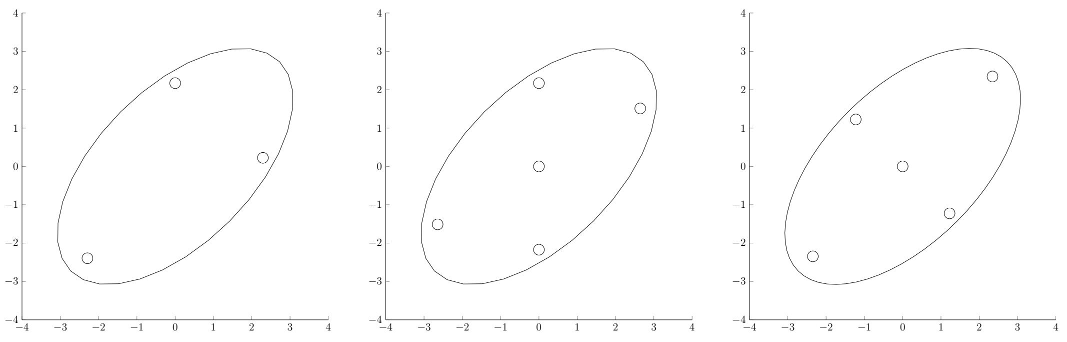

(a) Simplex (b) Cholesky (c) Eigenvectors

Figure 2.8: Three different ways of choosing a set of sigma points.

Figure 2.8 exemplifies the three most common methods for selecting sigma points: when the limits imposed on computational resources are stringent, or no information is given about the prior distribution other that its first two moments, the first method (Fig. 2.8a) defines a minimal simplex of $m + 1$ points on the m-hypersphere centered around the mean [13]. These sigma points are placed asymmetrically, though, and so cannot accurately reconstruct the skew (third order moment) and kurtosis (fourth order moment) when the propagation functions are known to exhibit some sort of symmetry.

With a slight increase in computational complexity, skewness is respected by sampling a set of $O(2m)$ points centered around the mean and defined as the rows of the square root of the covariance matrix; when kurtosis is known, such as in the case of prior Gaussian distributions, an additional point placed on the mean completes the set and guarantees an optimal behavior [16]. Figg. 2.8b and 2.8c show two equivalent sigma point sets that differ only in how $\sqrt{\mathbf{P}}$ was computed, respectively by the (numerically stable) Cholesky factorization algorithm $\mathbf{P} = \mathbf{L}\mathbf{L}^T$ and by deriving eigenvalues and eigenvectors $\mathbf{P} = \mathbf{V}\mathbf{D}\mathbf{V}^T$. For this thesis, Cholesky decomposition was preferred.

Finally, to capture more information it's always possible to increase the number of sigma points, e.g. to $O(2m^2 + 1)$ as [17] did. As is often the case, a balance must be struck between accuracy and usability.

The last thing to consider is how to *scale* the sigma points with respect to the covariance matrix. When large noise factors are present, the radius of the sphere that bounds the sigma points increases as well, and non-local effects may be sampled [14]: even though the overall information about the distribution is captured correctly, the filter may be inefficient.

Three scaling parameters α, β, κ are then defined:

- α typically resides in the interval $(0, 1]$, and determines the spread of the sigma points around the distribution mean. The value commonly used in literature [14] [13] [43], and the value used by the UKF developed in this thesis, is 1e-3.
- β expresses information about the shape of the prior distribution: for multivariate Gaussians, $\beta = 2$ gives optimal results and, in fact, a slightly conservative estimate of the posterior distribution.
- κ is a secondary parameter, used as a second degree of freedom to fine-tune the higher order moments. A good heuristic [15] is to set $m + \kappa = 3$, but negative values of κ might lead to a non-positive definite posterior covariance: common practice is, then, to set it to 0.

Referencing the augmented state 2.31 where $m = 2N' + 6$,[2] the final set of sigma vectors will be:

$$\begin{aligned} \mathcal{X}_0 &= \hat{\mathbf{x}}_a \\ \mathcal{X}_i &= \hat{\mathbf{x}}_a + c\left(\sqrt{\mathbf{P}_a}\right)_i \quad i = 1, \ldots, m \\ \mathcal{X}_{-i} &= \hat{\mathbf{x}}_a - c\left(\sqrt{\mathbf{P}_a}\right)_i \quad i = 1, \ldots, m \end{aligned} \tag{2.32}$$

[2] Two coordinates for each known node, three robot pose variables and three noise factors.

where

$$c = \sqrt{\lambda + m} \tag{2.33}$$

$$\lambda = \alpha^2(m + \kappa) - m \tag{2.34}$$

and $\left(\sqrt{\mathbf{P}_a}\right)_i = \mathbf{L}_i$ is the i-th row of the Cholesky factorization of the augmented state covariance.

To each sigma point $\mathcal{X}_i$ of the set 2.32 are then associated two corresponding weights $\mathcal{W}_i^{(m)}$ and $\mathcal{W}_i^{(c)}$; they are used when performing the inverse Unscented Transformation to recompute, respectively, the posterior mean and covariance. It's important to note that when α is small, the weights $W_0^{(m)}$ and $W_0^{(c)}$ are negative: this should not be a source of confusion, because the set of sigma points is *not* a probability distribution.

$$W_0^{(m)} = \frac{\lambda}{\lambda + m} \qquad W_{\pm i}^{(m)} = \frac{1}{2(\lambda + m)} \tag{2.35}$$

$$W_0^{(c)} = \frac{\lambda}{\lambda + m} + (1 - \alpha^2 - \beta) \qquad W_{\pm i}^{(c)} = 1/(2p\alpha^2) \tag{2.36}$$

2.5.3 Classical UKF

Prediction

Predicting how the state estimate will evolve from instant $t-1$ to instant t is done in a manner alike to how the EKF does it; that is, the sigma points derived from the augmented prior $\hat{\mathbf{x}}_a^{t-1|t-1}$ are propagated through the nonlinear motion function f in (2.20):

$$\bar{\mathcal{X}}_i^{t|t-1} = f(\bar{\mathcal{X}}_i^{t-1|t-1}, \mathbf{u}_{t-1}, \tilde{\mathcal{X}}_i^{t-1|t-1}) \tag{2.37}$$

where $\bar{\mathcal{X}}_i$ stands for the first $1 \ldots 2N' + 3$ terms of each sigma vector $\mathcal{X}_i$, corresponding of course to the original full state estimate, and $\tilde{\mathcal{X}}_i$ is used to represent the third-to-last and second-to-last entries, relative to the odometric noises ν_θ and

ν_D. This breakdown is made possible by the a priori independence between state and noises, i.e. by the **0** matrices appearing in the expression of the augmented covariance (2.31).

After propagation, it's possible to reconstruct the predicted unaugmented state estimate and covariance $\hat{\mathbf{x}}_{t|t-1}$ and $\mathbf{P}_{t|t-1}$, with the same notation and meaning as that of (2.24) and (2.25). This is achieved via the inverse Unscented Transform, consisting in nothing more than a weighted sample mean and a weighted sample covariance computation:

$$\hat{\mathbf{x}}_{t|t-1} \approx \sum_{i=-m}^{m} \mathcal{W}_i^{(m)} \bar{\mathcal{X}}_i^{t|t-1} \tag{2.38}$$

$$\mathbf{P}_{t|t-1} \approx \sum_{i=-m}^{m} \mathcal{W}_i^{(c)} \left[\bar{\mathcal{X}}_i^{t|t-1} - \hat{\mathbf{x}}_{t|t-1}\right] \left[\bar{\mathcal{X}}_i^{t|t-1} - \hat{\mathbf{x}}_{t|t-1}\right]^T \tag{2.39}$$

Update

The scope of the set of $2m+1$ propagated sigma points is not limited to prediction alone: in fact, the measurement function (2.21) associated to the update phase can be perfectly applied on each $\mathcal{X}_i$, as with the case of motion:

$$\mathcal{Z}_i^t = h(\bar{\mathcal{X}}_i^{t|t-1}, \tilde{\bar{\mathcal{X}}}_i^{t|t-1}) \tag{2.40}$$

where $\bar{\mathcal{X}}_i$ has the same meaning as before and $\tilde{\bar{\mathcal{X}}}_i$ is the last term of each sigma point, corresponding to the univariate Gaussian measurement noise parameter $\boldsymbol{\Psi}$.

The resulting set $\mathcal{Z}^t$ is comprised by the same number of $2m+1$ *regression points* [23], whose length p is proportional to the number of measurements performed during the current time step t, that is the number of nodes within communication range of the MB. Indeed, the inverse Unscented Transform can be applied to them, taking care to maintain the original weightings (2.35) and (2.36): this operation reconstructs the predicted measurement's mean and variance, as well as its covariance with each node's position:

$$\bar{\mathbf{z}}_t \approx \sum_{i=-m}^{m} \mathcal{W}_i^{(m)} \mathcal{Z}_i^t \tag{2.41}$$

$$\mathbf{P}_{\mathbf{z}} \approx \sum_{i=-m}^{m} \mathcal{W}_i^{(c)} \left[\mathcal{Z}_i^t - \bar{\mathbf{z}}_t\right] \left[\mathcal{Z}_i^t - \bar{\mathbf{z}}_t\right]^T \tag{2.42}$$

$$\mathbf{P}_{\mathbf{xz}} \approx \sum_{i=-m}^{m} \mathcal{W}_i^{(c)} \left[\bar{\mathcal{X}}_i^{t|t-1} - \hat{\mathbf{x}}_{t|t-1}\right] \left[\mathcal{Z}_i^t - \bar{\mathbf{z}}_t\right]^T \tag{2.43}$$

Finally, the unscented analogue to the EKF Kalman gain (2.28) can be computed; the posterior estimate mean and covariance are updated with the same formulae employed by the EKF, using the information contained within the innovation $z_t - \bar{\mathbf{z}}_t$, where z_t is the actual result of the measurement phase:

$$\mathbf{K} = \mathbf{P}_{\mathbf{xz}} \mathbf{P}_{\mathbf{z}}^{-1} \tag{2.44}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}(z_t - \bar{\mathbf{z}}_t) \tag{2.45}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}\mathbf{P}_{\mathbf{z}}\mathbf{K}^T \tag{2.46}$$

It's worth reminding that both $\hat{\mathbf{x}}_{t|t}$ and $\mathbf{P}_{t|t}$ belong to the unaugmented state space, since the information provided by the noise terms is totally embedded in the output of the functions (2.37) and (2.40).

2.5.4 SLAM-specific UKF

The acute reader will have already noted that in order to extract the sigma points from the augmented state, the UT requires to compute the square root of $\mathbf{P}_a$ at each temporal step of the filter algorithm: the computational cost of this operation is *cubic* in the size of the state vector, and may prove to be a very dangerous bottleneck for the relatively lightweight processor aboard the MB.

Fortunately, Huang et al. [10] derived a formulation of the UKF that's specifically designed around the SLAM context: starting from the observation that each motion or measurement operation involves only a small subset of the whole state, whose size does not depend from the state estimate size N', a correspondingly sized positive definite submatrix of $\mathbf{P}$ can be extracted such that the necessary sigma points are computed within constant time.

Prediction in $O(m)$

During propagation, only the robot pose and the (noisy) input odometry participate in the process model. Therefore, a reduced computational complexity is achievable by defining a smaller augmented state:

$$\hat{\mathbf{x}}_\alpha = \begin{bmatrix} \hat{\mathbf{x}}_r \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{P}_\alpha = \begin{bmatrix} \mathbf{P}_r & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_\theta^2 & 0 \\ \mathbf{0} & 0 & \sigma_D^2 \end{bmatrix} \tag{2.47}$$

Under the previous assumptions about the robot pose representation and the odometry input size, the dimension of this state vector is 5 at any given instant: the computational costs to obtain a set of sigma points are then constant in both time and space, as this set is very small. Without rewriting the same equations, we summarize the Unscented Transform of (2.32) with:

$$\mathcal{X}_{\alpha,i}^{t-1|t-1} = UT(\hat{\mathbf{x}}_\alpha^{t-1|t-1}, \mathbf{P}_\alpha^{t-1|t-1}) \qquad i = 0, \pm 1, \ldots, \pm 5 \tag{2.48}$$

The actual propagation through the function f is identical to its classical UKF counterpart: when updating the whole covariance, though, particular care must be given to elements not residing on the main block diagonal. In fact, while the new robot covariance $\mathbf{P}_r^{t|t-1}$ is reconstructed from the covariance of the sigma points, and while the network's covariance $\mathbf{P}_{i,j}^{t|t-1} = \mathbf{P}_{i,j}^{t-1|t-1}$ for $i \neq j$ and $i, j \in 1, \ldots, N'$ remains unchanged, robot-node covariances $\mathbf{P}_{r,n} = \mathbf{P}_{n,r}^T$ must still be appropriately corrected.

The solution resides in the *inferred* propagation Jacobian matrices implicitly defined by the unscented regression of f [11]:

$$\bar{\mathbf{P}}_{r,\alpha}^{t|t-1} \left(\mathbf{P}_\alpha^{t-1|t-1}\right)^{-1} = \begin{bmatrix} \breve{\mathbf{F}}_r^{t-1} & \breve{\mathbf{G}}_r^{t-1} \end{bmatrix} \tag{2.49}$$

where

$$\bar{\mathbf{P}}_{r,\alpha}^{t|t-1} \approx \sum_{i=-5}^{5} \mathcal{W}_i^{(c)} \left[\bar{\mathcal{X}}_{\alpha,i}^{t|t-1} - \hat{\mathbf{x}}_r^{t|t-1}\right] \left[\mathcal{X}_{\alpha,i}^{t-1|t-1} - \hat{\mathbf{x}}_\alpha^{t-1|t-1}\right]^T \tag{2.50}$$

and as before, each $\bar{\mathcal{X}}_{\alpha,i}^{t|t-1}$, with $i = 0, \pm 1, \dots, \pm 5$, is a vector containing the first three terms of the propagated sigma points $f(\bar{\mathcal{X}}_{\alpha,i}^{t-1|t-1}, \mathbf{u}_{t-1}, \tilde{\mathcal{X}}_{\alpha,i}^{t-1|t-1})$. In order to compute the cross-correlation between the predicted robot state and the nodes, it should be noted that:

$$\begin{aligned}
\mathbf{P}_{r,n}^{t|t-1} &= \mathrm{E}\left[\mathbf{x}_r^{t|t-1} \left(\mathbf{x}_n^{t-1|t-1}\right)^T\right] \\
&= \mathrm{E}\left[\left(\breve{\mathbf{F}}_r^{t-1}\mathbf{x}_r^{t-1|t-1} + \breve{\mathbf{G}}_r^{t-1}\begin{bmatrix}\nu_\theta^{t-1} \\ \nu_D^{t-1}\end{bmatrix}\right)\left(\mathbf{x}_n^{t-1|t-1}\right)^T\right] \\
&= \breve{\mathbf{F}}_r^{t-1}\mathbf{P}_{r,n}^{t-1|t-1}
\end{aligned} \tag{2.51}$$

Finally, the predicted state covariance will be:

$$\mathbf{P}_{t|t-1} = \begin{bmatrix} \mathbf{P}_r^{t|t-1} & \breve{\mathbf{F}}_r^{t-1}\mathbf{P}_{r,n}^{t-1|t-1} \\ \mathbf{P}_{n,r}^{t-1|t-1}\left(\breve{\mathbf{F}}_r^{t-1}\right)^T & \mathbf{P}_n^{t-1|t-1} \end{bmatrix} \tag{2.52}$$

The computational costs of this improved prediction phase are $O(1)$ for the direct and inverse Unscented Transformation, due to the fixed size of $\mathbf{P}_\alpha$, and $O(2N')$ for the predicted robot-node covariance computation. With the exception of a little overhead, this is comparable to the EKF performance.

Update in $O(m^2)$

The same line of reasoning can be adopted when updating the state estimate: at each time instant t, the MB can sense only a small number of nodes comprising its local neighborhood, defined as the intersection between the operative area and the communication radius of the robot's antenna. By filtering one measure at a time, the only state variables involved are the robot's position (regardless of its heading) and the relative n-th node position. The set of sigma points will then be drawn from the reduced augmented state:

$$\hat{\mathbf{x}}_\beta = \begin{bmatrix} \hat{\mathbf{x}}_r \\ \hat{\mathbf{x}}_n \\ 0 \end{bmatrix}, \quad \mathbf{P}_\beta = \begin{bmatrix} \mathbf{P}_r & \mathbf{P}_{r,n} & \mathbf{0} \\ \mathbf{P}_{n,r} & \mathbf{P}_n & 0 \\ \mathbf{0} & 0 & \sigma_\Psi^2 \end{bmatrix} \tag{2.53}$$

Again, the size of the covariance matrix is fixed at 5 elements, so that taking its square root is a $O(1)$ operation. Keeping the previous notation, both UT and RSSI measurement can be written as:

$$\mathcal{X}_{\beta,i}^{t|t-1} = UT(\hat{\mathbf{x}}_\beta^{t|t-1}, \mathbf{P}_\beta^{t|t-1}) \qquad i = 0, \pm 1, \ldots, \pm 5 \tag{2.54}$$

and

$$\mathcal{Z}_i^t = h\left(\bar{\mathcal{X}}_\beta^{t|t-1}, \tilde{\mathcal{X}}_\beta^{t|t-1}\right) \tag{2.55}$$

The *inferred* measurement Jacobian has an analogue role to (2.49) as a regression matrix for the generic LRKF. As the distance measurement projects the augmented state space to a single dimension, this matrix is shaped as a row vector:

$$\mathbf{P}_{\mathbf{z}\beta}^{t|t-1}\left(\mathbf{P}_\beta^{t|t-1}\right)^{-1} = \begin{bmatrix} \breve{\mathbf{H}}_r^t & \breve{\mathbf{H}}_n^t & \breve{\mathbf{H}}_\Psi^t \end{bmatrix} \tag{2.56}$$

where the submatrices on the right side correspond, respectively, to the robot position, the node position, and the noise term. Additionally, using formula (2.41) to derive the expected measure $\bar{\mathbf{z}}$, the covariance between observations and augmented state is:

$$\mathbf{P}_{\mathbf{z}\beta}^{t|t-1} \approx \sum_{i=-5}^{5} \mathcal{W}_i^{(c)} \left[\mathcal{Z}_i^t - \bar{\mathbf{z}}_t\right] \left[\mathcal{X}_{\beta,i}^{t|t-1} - \hat{\mathbf{x}}_\beta^{t|t-1}\right]^T \tag{2.57}$$

The matrix constructed at (2.56) cannot be applied yet to update the posterior state and covariance, as it relates to $\hat{\mathbf{x}}_\beta$: discarding the entry relative to the measurement noise and zero-padding the terms corresponding to unobserved nodes, the full inferred Jacobian becomes:

$$\breve{\mathbf{H}}_t = \begin{bmatrix} \breve{\mathbf{H}}_r^t & \mathbf{0} & \ldots & \mathbf{0} & \breve{\mathbf{H}}_n^t & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix} \tag{2.58}$$

with $\breve{\mathbf{H}}_n^t$ assuming the same position in $\breve{\mathbf{H}}$ as the node n is indexed in the state vector $\hat{\mathbf{x}}$. Finally, the posterior is updated via the standard Kalman equations for a linear system:

$$\mathbf{K} = \mathbf{P}_{t|t-1}\breve{\mathbf{H}}_t\mathbf{P}_{\mathbf{z}}^{-1} \tag{2.59}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}(z_t - \bar{\mathbf{z}}_t) \tag{2.60}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}\mathbf{P}_{\mathbf{z}}\mathbf{K}^T \tag{2.61}$$

Being once again dominated by the covariance update in (2.61), the overall complexity for *each* measurement update is quadratic in the number of nodes, i.e. an order of magnitude faster than its classical counterpart. Assuming that M known nodes are sensed in a single time step, this results in a workload of $O(MN'^2)$ operations; when the communication range is large enough (or close to large enough) to cover the whole operating area, M tends to N and a cubic time complexity may be inevitable. In such a case where no performance gain is achieved over a standard UKF, it would still be preferable for each batch of measures $\mathbf{z}(t)$ to be incorporated one at a time, as the filtering algorithm can execute in parallel with the RSSI data collection.

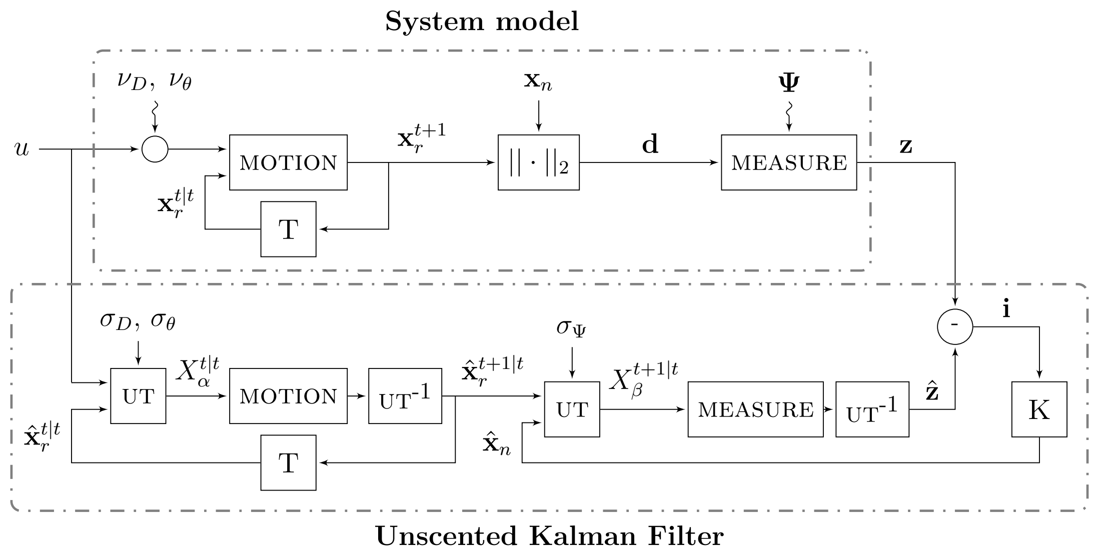


Figure 2.9: Block model of an Unscented Kalman Filter for SLAM.

2.6 Stopping criteria

The recursive formulae employed by the Kalman Filter, be it Extended or Unscented, strive to continuously improve the WSN nodes localization by aggregating the information from a potentially endless stream of measurements. Of course, taking into account WSN and MB battery consumption, eventual distance and time requirements, and the *information submodularity*[3] [22] property inherent to the measurement process, there must be a set of conditions that when verified can stop the filtering algorithm.

The first such condition is *WSN coverage*: the total number of nodes N deployed over the operative area is assumed to be known a priori to the algorithm, which will not stop until that same number of nodes has been initialized into the state estimate $\hat{\mathbf{x}}$, that is until $N = N'$.

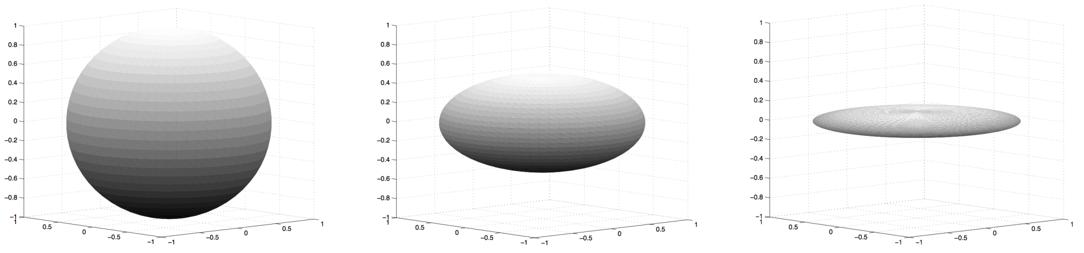
This requirement, though, only works for networks which are fully static, both in composition and location. To provide a counterexample, the same nodes may change their own position within the operating area according to some stochastic (possibly Markovian) process, as books do when moving between shelves and tables in a library: in such as a case, the AMR may need to cyclically start a new localization process after a certain time interval, still being subject to the same condition on coverage. In another hypothetical scenario, the number of nodes may not be a constant quantity, possibly due to battery failures, or node removal and insertion: for example, smart goods which are loaded and unloaded in a warehouse. In similar situations, area coverage and/or continuous patrolling may be better planning choices.

The second main halting condition is, quite obviously, related to how well the

[3] Intuitively, this property can be described as a law of diminishing returns: making the MB perform a RSSI range measurement when the set $\mathcal{M}$ of past measurements is small brings more overall information than when $\mathcal{M}$ is larger. This monotonicity is strict when measuring a node already discovered.

nodes are localized: since there is no way for the mobile robot to evaluate accuracy without also knowing the ground truth (and thus defeating the whole purpose of WSN localization), a measure of precision will offer the next best alternative. In other words, the localization algorithm should stop when a certain function of the covariance matrix $s(\mathbf{P})$ crosses a given confidence threshold, which usually depends on the specific implementation's requirements.

Searching for such a function should be easier when considering that it does not need to be applied to the whole covariance; instead, knowing how each WSN node's position is represented in the state $\mathbf{x}$, one could define a reduced covariance matrix $\mathbf{P}_n$, with $n = 1, \dots, N$, as the 2-by-2 block residing on the main diagonal of $\mathbf{P}$ and corresponding to node n. It's simpler then to evaluate N times a smaller function $s'(\mathbf{P}_n)$, and to combine the result in a logical conjunction: if any of those nodes is not yet localized, the algorithm should continue its execution.



(a) All eigenvalues at 1
$\det \mathbf{A} = 1$
$\mathrm{Tr}(\mathbf{A}) = 3$

(b) One eigenvalue at 0.5
$\det \mathbf{A} = 0.5$
$\mathrm{Tr}(\mathbf{A}) = 2.5$

(c) One eigenvalue at 0.1
$\det \mathbf{A} = 0.1$
$\mathrm{Tr}(\mathbf{A}) = 2.1$

Figure 2.10: Three uncertainty ellipses for the example covariance matrix $\mathbf{A}$.

The two main candidates for $s'(\cdot)$ are the determinant and the trace functions. Both express some function of the eigenvalues λ_i of a square matrix $\mathbf{A}$ of size m, one being their productory and one being their sum:[4]

[4] Borrowing two definitions from the field of experiment design, one could also speak of *d-halting* and *a-halting* criteria.

$$\det(\mathbf{A}) = \prod_{i=1}^{m} \lambda_i \tag{2.62}$$

$$\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{m} \lambda_i \tag{2.63}$$

Choosing one function over the other is often related to the problem at hand: as Roy and Sim argued [36], each of those functionals capture a different geometric property of the hypersphere that bounds the estimate uncertainty.

The determinant is a measure of (although not equal to) the volume of said hypersphere, and while being invariant to the scale of the state variables it possesses a disadvantageous property: namely, it's possible to drive $\det(\mathbf{P}_n)$ to 0 by reducing a single eigenvalue to 0, making the matrix singular. A trivial way to do so is to move the MB in such a pattern that only information about one direction is acquired, disregarding its orthogonal: when the node position distribution would be represented by an uncertainty ellipse, a characteristic squeezed shape would be observed.

For a localization problem, though, the same units and scale are shared by all the variables in the state space, with the exception of the robot heading parameter θ which can be safely ignored: this overcomes the main limitations of trace (scaling, physical meaning) as a confidence metric. As Fig. 2.10 exemplifies, a geometrical interpretation of trace would be the total sum of the axes of the confidence hyperellipses: this "average" uncertainty is considered to be more robust and more apt to capture the overall uncertainty of the model.

Chapter 3

Node initialization

UNLESS some sort of a priori knowledge is available about the initial WSN spatial configuration, the localization filter has no idea how to assign a starting position to each node, not even in a rough sense. Indeed, in some of the contexts specified in the previous examples, it may not even be *aware* of the presence of an unlocalized node! Since the Kalman recursive formulae are only capable of refining an already existing state estimate, the problem of initializing new state variables in the filter must be solved separately.

The event of discovering a new node can be easily summarized: during the whole execution of the RUUMBA procedure, the Mobile Beacon maintains an incremental key/value data structure (e.g. an indexed array, or a hashed dictionary) which stores each node's current position estimate $\hat{\mathbf{x}}_n$. Each entry is indexed by an unique identifier, characteristic to each WSN mote and included in the header of every transmitted message: it could be a name or a network address assigned by the MB as in [3], the physical MAC address of the device, and so on. This identifier is used to perform the task of data association between each message's sender and the ordinal position of its state variables within vector $\hat{\mathbf{x}}$; when there is no match between an incoming identifier and the list of keys, the MB updates the state estimate by appending a new value $\hat{\mathbf{x}}_{key}$. The exact shape and nature of this prior distribution depend on how this initialization is performed.

3.1 Naive initialization

A valid yet extremely approximate solution would be that of making no assumptions about the node initial position other than the fact that it's within the deployment area: effectively, this translates into an initial bivariate uniform distribution defined over the whole area, eventually approximable to a very wide and flat Gaussian distribution. As a slightly more refined solution, the support could be intersected with the area within the communication radius defined by the antenna sensitivity. Still, this approach is insufficient: as can be seen in Fig. 3.1, neither the mean (red triangle) nor the covariance (represented by the red square corresponding to a 95% confidence interval) of the node location estimate depend on the distance measurement that triggers the initialization, whose information then goes unused.

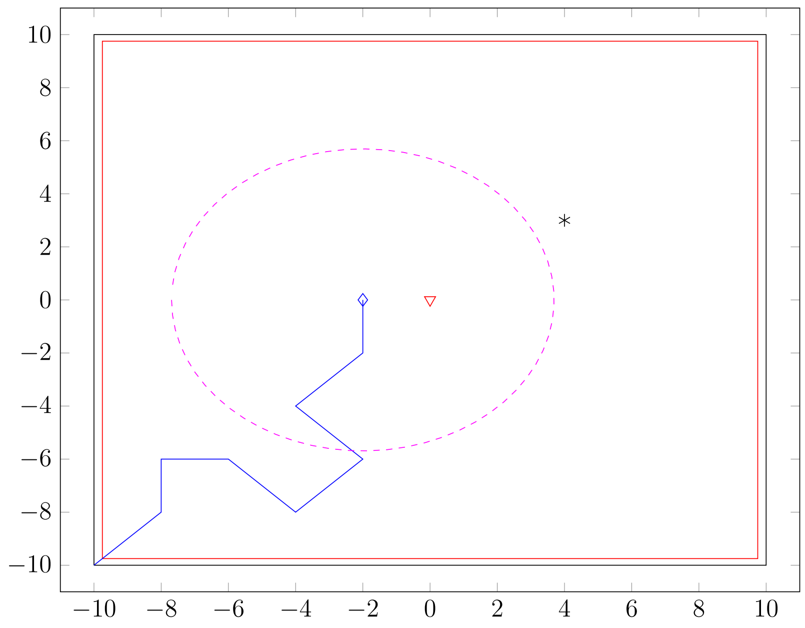


Figure 3.1: Example of a naive initialization.

Moreover, Kalman localization is quite sensitive to initial conditions, even if recursively improved by subsequent measures [29]: therefore, this kind of initialization tries too much to minimize the risk of an initial outlier, with the result of worsening the overall performance.

3.2 Multilateration

To improve the starting localization, some constraints have to be imposed. From a purely geometrical standpoint, even in the absence of noise a single range measurement is insufficient to identify a single point of origin for the received transmission. Rather, a delay is introduced as communication must be attempted by the MB from multiple vantage points, recording the robot position for each deferred observation [1]. When $m \geq 3$ RSSI range measurements $\tilde{d}_1, \ldots, \tilde{d}_m$ have been collected from as many non-collinear beacon locations $\mathbf{x}_1^r, \ldots, \mathbf{x}_m^r$, a sufficient quantity of information has been obtained and node initialization can proceed.

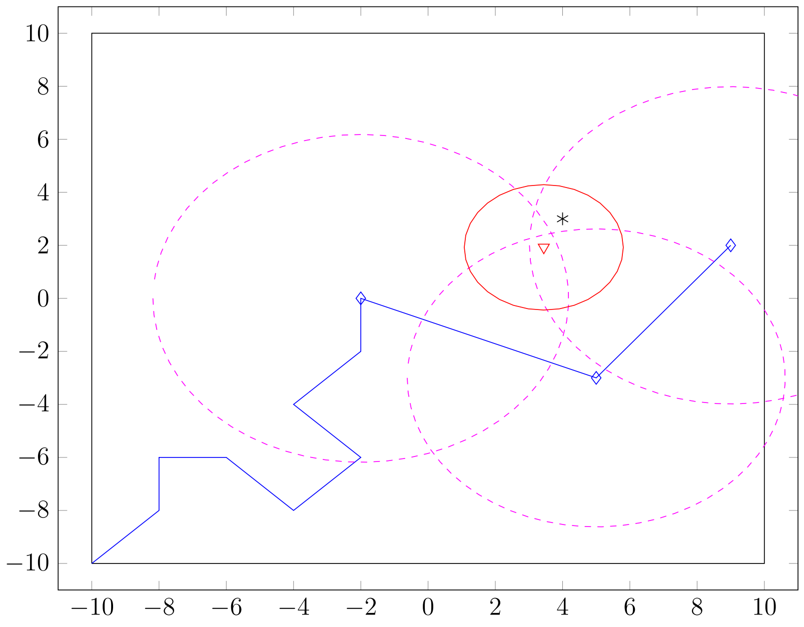


Figure 3.2: Example of a multilateration initialization, with m=3.

Least squares

If observations were noiseless, finding the best node position that fits m constraints would be simply a matter of choosing 3 random distance measurements and trilaterating; unfortunately, noise disturbances may cause the circumferences with radii equal to the estimated distances to not intersect in a single point (cfr. Fig. 3.2).

One of the simplest way to get the best average position, then, is to define a least square problem consisting of $m-1$ equations over the 2 planar coordinates x_n, y_n of the discovered node:

$$\mathbf{A}_n \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{2}\mathbf{b}_n \tag{3.1}$$

$$\begin{bmatrix} (\hat{\mathbf{x}}_1^r - \hat{\mathbf{x}}_m^r)^T \\ \vdots \\ (\hat{\mathbf{x}}_{m-1}^r - \hat{\mathbf{x}}_m^r)^T \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{2} \begin{bmatrix} (\tilde{d}_m^2 - \tilde{d}_1^2) + ||\hat{\mathbf{x}}_1^r||^2 - ||\hat{\mathbf{x}}_m^r||^2 \\ \vdots \\ (\tilde{d}_m^2 - \tilde{d}_{m-1}^2) + ||\hat{\mathbf{x}}_{m-1}^r||^2 - ||\hat{\mathbf{x}}_m^r||^2 \end{bmatrix} \tag{3.2}$$

$$\begin{bmatrix} \hat{x}_n \\ \hat{y}_n \end{bmatrix} = \frac{1}{2}(\mathbf{A}_n^T\mathbf{A}_n)^{-1}\mathbf{A}_n^T\mathbf{b}_n \tag{3.3}$$

The starting covariance can be the identity matrix $\mathbf{I}_2$, corresponding to a circular normal distribution, appropriately scaled by the average of all the range measurement variances [29].

Temporarily augmented state

Another approach is to exploit the already available SLAM logic, and to augment the state vector by including the collected robot poses [1]:

$$\hat{\mathbf{x}}_{init} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{x}}_1^r \\ \vdots \\ \hat{\mathbf{x}}_m^r \end{bmatrix} \tag{3.4}$$

Correspondingly, the measurements performed at each of these poses are stored in an auxiliary list $\{\tilde{d}_1, \ldots, \tilde{d}_m\}$; after constructing, via a batch update, the estimate mean and covariance, these additional terms can be safely discarded and the state returned to its original formulation.

3.3 Particle Filtering

Lateration effectiveness, though, depends heavily on the mean values of the measurements used as constraints, with no regard to their variance: as a result, all observations are weighted equally and the occasional outlier has to be averaged over a large m. In truth, a simple circumference is not an adequate representation, and should be disparaged in favor of an annulus shape, centered around the mean and with radial distribution that is directly proportional to the range observation, i.e. highest at the measured distance $\tilde{d}$ and sloping inward and outward according to the lognormal formula (2.8). The annuli intersection distribution, which of course is the (possibly multimodal) Bayesian combination of each annular distribution, eventually can be approximated into a bivariate Gaussian that includes the sensed node.

Since it's impossible to analytically model these distributions, a Monte Carlo particle filter is instantiated for each discovered node, as in [28]. At first, a large number P of particles is generated randomly from the mutually independent polar coordinates distributions centered on the MB location; successively, for ease of computation, they are converted to their Cartesian equivalents:

$$r_p \sim \log\mathcal{N}\left(\ln\tilde{d}, \left(\tfrac{\ln 10}{10\eta}\sigma_\Psi\right)^2\right) \tag{3.5}$$

$$\phi_p \sim \mathcal{U}\left(0, 2\pi\right) \tag{3.6}$$

$$\mathbf{x}_p = \begin{bmatrix} r_p\cos\phi_p & r_p\sin\phi_p \end{bmatrix} \tag{3.7}$$

The initial importance weight w_p associated with each particle $p = 1, \ldots, P$ is determined by its likelihood given the measure $\tilde{d}$. Ensuing distance estimations modify these weights through Bayesian update and normalization:

$$w_p^{t+1} = \frac{w_p^t \cdot \ell\left(\tilde{d}_t \mid \mathbf{x}_p\right)}{\sum_{p=1}^{P} w_p^t \cdot \ell\left(\tilde{d}_t \mid \mathbf{x}_p\right)} \tag{3.8}$$

A *Sampling Importance Resampling* (SIR) algorithm is then applied each time a new observation is available, to avoid filter degeneracy. Simply put, when the estimated number of effective particles $N_{eff} = 1/\sum_{p=1}^{P}(w_p)^2$ is below a certain threshold, a whole new set of P particles is extracted with replacement from the current one, and the importance weights are reinitialized as $w_p = 1/P$.

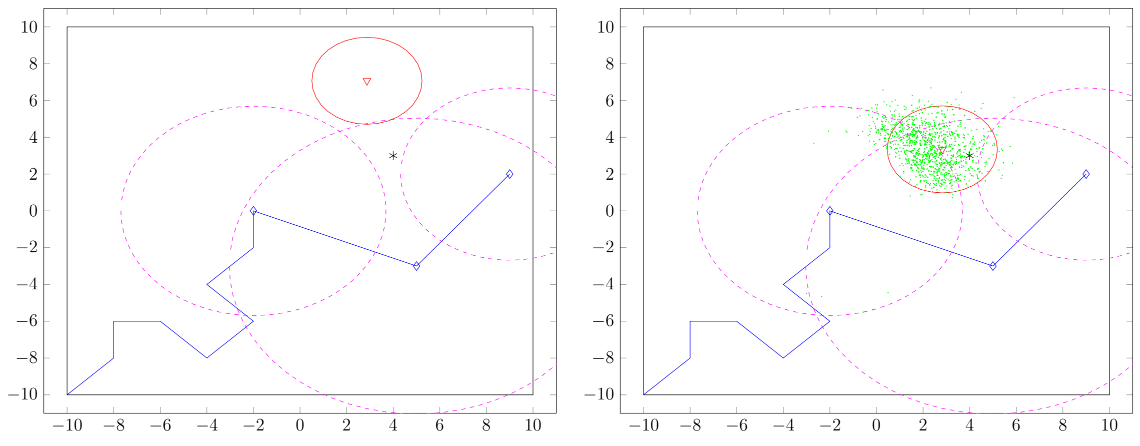

(a) Multilateration (b) Particle filtering

Figure 3.3: Comparison between a multilateration initialization and a particle filter initialization when noisy measurements are present.

Finally, when the particle set has converged below a certain level of spatial sparseness, i.e. when some statistic of its covariance (such as those specified in Sec. 2.6) is under a given threshold, delayed initialization is over and the whole set can be condensed into a Gaussian having the same first two moments.

The effectiveness of this initialization technique can be seen in Fig. 3.3: like all Monte Carlo approximations, the closeness to the real posterior distribution is related to the number P of particles used for each filter. Remembering that this $O(P)$ complexity cost must be beared for the entire length of the delay and for each node currently known yet unlocalized, this makes the recourse to particle filters a potent but very resource demanding approach.

3.4 Gaussian Mixture Models

Looking back at section 2.5, it can be seen how the main reason behind the claimed efficiency superiority of the UKF is that a small, constant quantity of deterministically chosen points could approximate reasonably well any given posterior distribution. The same basic principle of *divide et impera* can be applied to the initialization problem [37]; in fact, the starting annulus distribution can be substituted with a corresponding *Gaussian Mixture Model* (GMM), consisting of a weighted sum of H bivariate normal distributions:

$$f_{\text{ANNULUS}}(\hat{\mathbf{x}}_n) \approx \sum_{h=1}^{H} w_h \, \mathcal{N}(\hat{\mathbf{x}}_{n,h}, \mathbf{P}_{n,h}) \tag{3.9}$$

where the nontrivial weights $0 < w_h < 1$ sum up to 1.

The ability of representing arbitrary distributions as a linear combination of Gaussians allows for a simpler integration into the whole class of Kalman filters. Indeed, analogously to the behavior of a sigma point, each of these Gaussian distributions can be "propagated" by including it in the state as an independent hypothesis about the node location: any new measurement would update both mean and covariance of *all* these H hypotheses $\hat{\mathbf{x}}_{n,h}$, according to the implemented SLAM filter. Since this aforementioned inclusion is performed right after the first measure, there is no time delay to be waited before the localization algorithm gets a general idea about the node's whereabouts.

The weighted sum of these transformed distributions, where the weights are updated similarly to (3.8), should reconstruct a similarly transformed multimodal posterior estimate; nonetheless, it's generally more efficient to just discard duplicate or unlikely hypotheses until the model converges to just one Gaussian. This last hypothesis, then, would assume the role of node position estimate $\hat{\mathbf{x}}_n$.

Following the work of [6] [30], an adaptation of this initialization method was created for the specific case of lognormal range measurements.

3.4.1 Annulus initialization

Approximating a node's location estimate distribution after just one distance measure $\tilde{d}$ is done in a way not unlike that of particle filters; that is, by defining a polar coordinate system around the current MB position estimate $\hat{\mathbf{x}}_r$ it can be seen that the radial coordinate distribution $\mathbf{r}_{n,h} = \mathbf{r}_n$ is shared by each hypothesis and keeps the expression already derived in (3.5), while the angular coordinate distribution $\mathbf{\Phi}_n = \mathbf{\Phi}$ is always a uniform distribution that can be decomposed into a mixture model:

$$\mathbf{\Phi} = \mathcal{U}(0, 2\pi) \approx \sum_{h=1}^{H} w_h \mathbf{\Phi}_h \tag{3.10}$$

$$\mathbf{\Phi}_h = \mathcal{N}(\phi_h, \sigma_{\phi,h}^2) \tag{3.11}$$

where the weights are all set to $w_h = 1/H$. Particular care must be given to the subtle difference between $\mathbf{\Phi}$ and a common uniform distribution, in that the former's support is actually the whole space $\mathbb{R}$ modulo 2π.

Exploiting the angular symmetry of the annulus, it's easy to see that the means ϕ_h will be evenly spaced around the whole circumference:

$$\phi_h = h \cdot \frac{2\pi}{H}, \qquad h = 1, \ldots, H \tag{3.12}$$

and the standard deviations $\sigma_{\phi,h}$ will all be equal to a single value σ_ϕ, the optimal value for which has been empirically calculated by [6] as:

$$\sigma_\phi = \frac{2\pi}{1.5H} \tag{3.13}$$

To test the above expression, several simulations have been made for H ranging from 2 to 20. The simulations showed that the model is a good approximation of the objective uniform distribution for $H \geq 6$.

The most accurate way to convert each hypothesis's mean and covariance from its polar coordinates representation (r_h, ϕ_h) to a Cartesian coordinate system is to apply the Unscented Transform. The sigma points are sampled from the Gaussian distributions $\mathbf{\Psi}$ and $\mathbf{\Phi}_h$, then transformed by a conversion function parametrized by the measured distance:

$$\hat{\mathbf{x}}_{n,h} = f(\tilde{d}_n, \boldsymbol{\Psi}, \boldsymbol{\Phi}_h) \tag{3.14}$$

$$= \tilde{d}_n \cdot 10^{\boldsymbol{\Psi}/10\eta} \cdot \begin{bmatrix} \cos \boldsymbol{\Phi}_h \\ \sin \boldsymbol{\Phi}_h \end{bmatrix} \tag{3.15}$$

Finally, their mean and covariance are recomputed through the inverse Unscented Transform formulae (2.38-2.39).

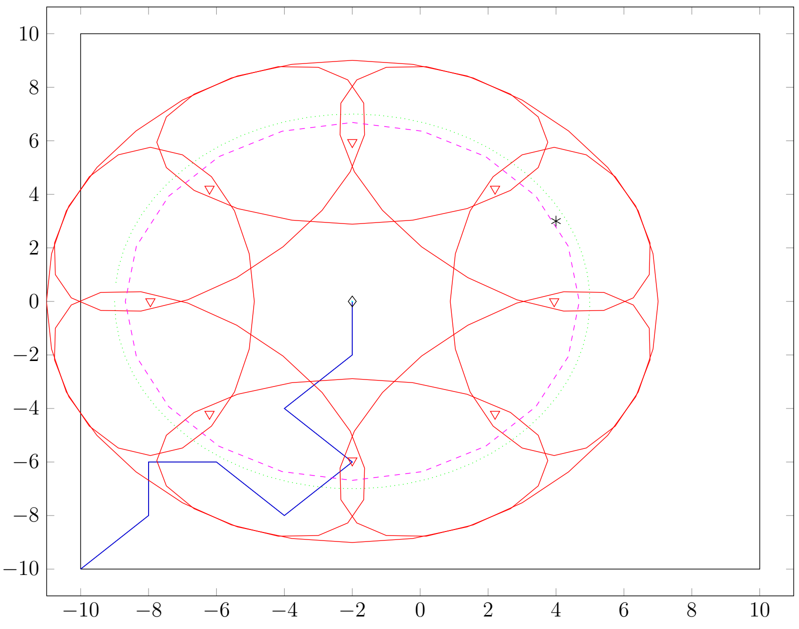
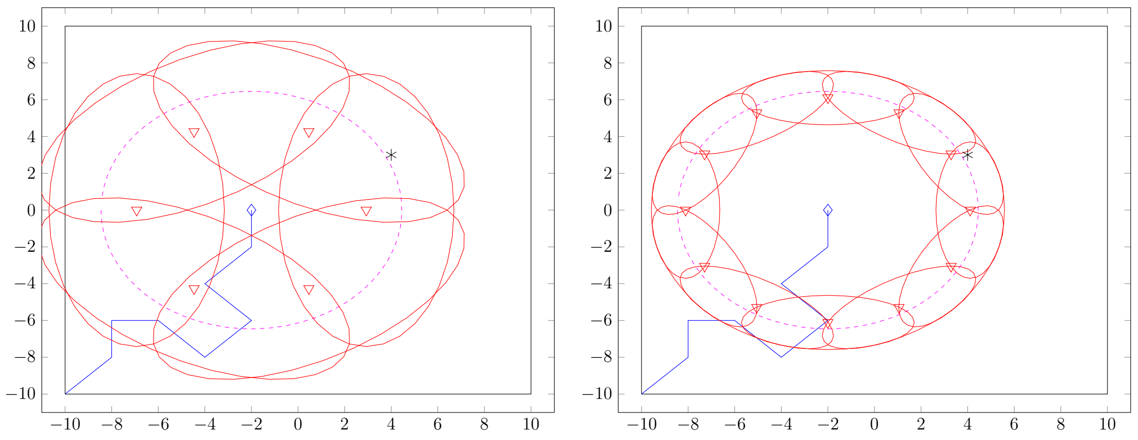


Figure 3.4: Annulus approximation with $H = 8$ hypotheses.

Figure 3.4 shows a graphical example of this initialization: the hypotheses' means are not located on the dashed magenta circumference with radius $\tilde{d}$, because due to the angular variance σ_ϕ the conversion from polar coordinates tends to be "banana-shaped" (cfr. Fig. 2.7). Moreover, as it would be reasonable to expect from a GMM, the 95% confidence ellipsoids overlap one another.

As a side note, the optimal choice of the number of hypotheses H will be discussed: it's immediately evident that, whenever a new node is discovered, $2H$ variables will be added to the filter state adding to the overall computational costs. Using a very large number of hypotheses would then be overkill, yet reducing this number too much could lead to inconsistencies. In fact, by visualizing each covariance $\mathbf{P}_{n,h}$ with its confidence interval, to improve the initial annulus approximation the length of each ellipse's radial axis (i.e. the radial variance of $\hat{\mathbf{x}}_{n,h}$ with regards to $\mathbf{x}_r$) should depend only by the range measurement variance. Reducing H leads to more pronounced "banana shapes", which shift the means inward and add unnecessary uncertainty.

Since RSSI range variance depends on the distance from the node, it could be possible to derive a formula to choose the optimal number of hypotheses depending on the value of $\hat{d}_n$. However, experimental results showed that $H = 8$ is in general a good enough choice for all practical ranges at which a node is discovered, and will be used for the rest of this thesis.



(a) H=6 (b) H=12

Figure 3.5: Example of the positive effect of increasing the number of hypotheses, for constant range estimation variance.

3.4.2 Incorporating measurements

Once the GMM has been initialized, following observations are used to update the estimate of each hypothesis and to refine the weights $w_{n,h}$ associated to them: still, a single measurement cannot be applied as it is, because that would mean exploiting H times the information contained within its variance $\mathrm{Var}[\hat{\mathbf{d}}_n]$, eventually leading to filter divergence.

Fortunately, as shown by [37], the estimate correction provided by a single measure of variance $\mathbf{R}$ is equivalent to that of a set of H measurements with same mean and whose variances $\mathbf{R}_h$ satisfy the following relation:

$$\mathbf{R}^{-1} = \sum_{h=1}^{H} \mathbf{R}_h^{-1} \tag{3.16}$$

This means that the original information can be partitioned into H new disjoint virtual measurements, each applied to a different hypothesis and with a smaller degree of informativeness.

A proper information splice should not be arbitrary, though, but should give more weight to (i.e. have more influence over) those hypotheses that agree with the measurement. It's useful then to define a convex set of weights $\lambda_{n,h}$ with $\sum_{h=1}^{H} \lambda_{n,h} = 1$, which are proportional to the normalized likelihoods of the measured distance $\tilde{d}_n$ given the expected distance $\hat{d}_{n,h} = ||\hat{\mathbf{x}}_{n,h} - \mathbf{x}_r||_2$ between each hypothesis and the MB [37]:

$$\lambda_{n,h} = \frac{\ell(\tilde{d}_n \,|\, \hat{d}_{n,h})}{\sum_{h=1}^{H} \ell(\tilde{d}_n \,|\, \hat{d}_{n,h})} \tag{3.17}$$

where the expression for $\ell(\tilde{d} \,|\, \hat{d})$ is given by (2.17).

It's often the case that, as the measurement noise variance increases, these likelihoods are too small to be computed without risking numerical underflow. To prevent such conditions, it's useful to translate the above formula to log-space:

$$\begin{aligned} \Lambda_{n,h} = \ln \lambda_{n,h} &= \ln \left(\frac{\ell(\tilde{d}_n|\hat{d}_{n,h})}{\sum_{h=1}^{H} \ell(\tilde{d}_n|\hat{d}_{n,h})} \right) \\ &= \mathcal{L}(\tilde{d}_n|\hat{d}_{n,h}) - \ln \left(\sum_{h=1}^{H} e^{\ln \ell(\tilde{d}_n|\hat{d}_{n,h})} \right) \\ &= \mathcal{L}(\tilde{d}_n|\hat{d}_{n,h}) - \ln \left(\sum_{h=1}^{H} e^{\mathcal{L}(\tilde{d}_n|\hat{d}_{n,h})} \right) \\ &= \mathcal{L}(\tilde{d}_n|\hat{d}_{n,h}) - A - \ln \left(\sum_{h=1}^{H} e^{\mathcal{L}(\tilde{d}_n|\hat{d}_{n,h}) - A} \right) \end{aligned} \tag{3.18}$$

where $\mathcal{L}(\tilde{d}|\hat{d})$ represents log-likelihoods, and in the last step the log-sum-exp trick was adopted using $A = \max_h \mathcal{L}(\tilde{d}_n \mid \hat{d}_{n,h})$.

Remembering that the expression for RSSI range estimation variance can be approximated by substituting the true distance d_{true} in formula (2.15) with the measurement $\tilde{d}_n$, it's possible to extract the new variance of each virtual observation:

$$\begin{aligned}\mathbf{R}_{n,h} &= \lambda_{n,h}^{-1} \cdot \mathbf{R}_n \\ &= e^{-\Lambda_{n,h}} \cdot \mathbf{R}_n \\ &= e^{-\Lambda_{n,h}} \cdot \tilde{d}_n^2 \cdot \left(e^{\sigma^2} - 1\right) \end{aligned} \tag{3.19}$$

Moreover, as seen in the previous chapter, the UKF does not draw its sigma points from the lognormal distribution $\hat{\mathbf{d}}$, but rather from the underlying Gaussian distribution of its noise factor $\boldsymbol{\Psi}$: it's therefore also useful to directly compute the modified variance $\sigma^2_{\Psi,n,h}$ which is to be integrated into the augmented covariance matrix $\mathbf{P}_\beta$ of equation (2.53):

$$\begin{aligned}\tilde{d}_n^2 \cdot \left(e^{\sigma^2_{n,h}} - 1\right) &= e^{-\Lambda_{n,h}} \cdot \tilde{d}_n^2 \cdot \left(e^{\sigma^2} - 1\right) \\ e^{\sigma^2_{n,h}} &= 1 + e^{-\Lambda_{n,h}} \cdot \left(e^{\sigma^2} - 1\right) \\ \sigma^2_{n,h} &= \ln\left(1 + e^{-\Lambda_{n,h}} \cdot \left(e^{\sigma^2} - 1\right)\right)\end{aligned} \tag{3.20}$$

With the aid of two successive Taylor approximations, the above formula can be simplified into:

$$\sigma^2_{n,h} = \ln\left(1 + e^{-\Lambda_{n,h}}\sigma^2 + O(\sigma^4)\right) \tag{3.21}$$

$$= e^{-\Lambda_{n,h}}\sigma^2 + O(\sigma^4) \tag{3.22}$$

$$\approx \frac{\sigma^2}{\lambda_{n,h}} \tag{3.23}$$

Given that σ and σ_Ψ differ only by a factor of $\frac{\ln 10}{10\eta}$, the proportion (3.23) stands for both.

Weight update

To each hypothesis h in a GMM is associated an importance weight $w_{n,h}$, that expresses the degree of belief by which the node is best estimated by that Gaussian distribution. As with the case of particle filtering, each distance measurement modifies those weights according to the normalized likelihoods computed before:

$$w_{n,h}^{t+1} = w_{n,h}^{t} \cdot \frac{\lambda_{n,h}^{t}}{\sum_{h=1}^{H} w_{n,h}^{t} \lambda_{n,h}^{t}} \tag{3.24}$$

Again, moving to the logarithmic form of these weights should help avoiding numerical errors:

$$\begin{aligned} \ln w_{n,h}^{t+1} &= \ln w_{n,h}^{t} + \Lambda_{n,h}^{t} - \ln\left(\sum_{h=1}^{H} w_{n,h}^{t} \lambda_{n,h}^{t}\right) && (3.25) \\ &= \ln w_{n,h}^{t} + \Lambda_{n,h}^{t} - \ln\left(\sum_{h=1}^{H} e^{\ln w_{n,h}^{t} + \Lambda_{n,h}^{t}}\right) && (3.26) \\ &= \ln w_{n,h}^{t} + \Lambda_{n,h}^{t} - B - \ln\left(\sum_{h=1}^{H} e^{\ln w_{n,h}^{t} + \Lambda_{n,h}^{t} - B}\right) && (3.27) \end{aligned}$$

where $B = \max_h(\ln w_{n,h}^{t} + \Lambda_{n,h}^{t})$.

A slightly different strategy should be adopted when a measurement is *not* received from a node already included in the state vector: in that case, the corresponding hypotheses that are closest to the MB location should see their weight decrease, because it's more unlikely that the message would be lost traveling a smaller distance. A graphical example is presented at Fig. 3.6, where the two least probable hypotheses are discarded from the system state.

The exact formula for updating the log-weights $\ln w_{n,h}$ hinges on the probability of receiving a message assuming the estimated distances $\hat{d}_{n,h}$, which is calculated as the Cumulative Distribution Functions of either the Gaussian received power with respect to a threshold on power sensitivity P_{th} of the beacon's antenna, or the lognormal distance estimate with respect to the robot's communication radius d_{comm}:

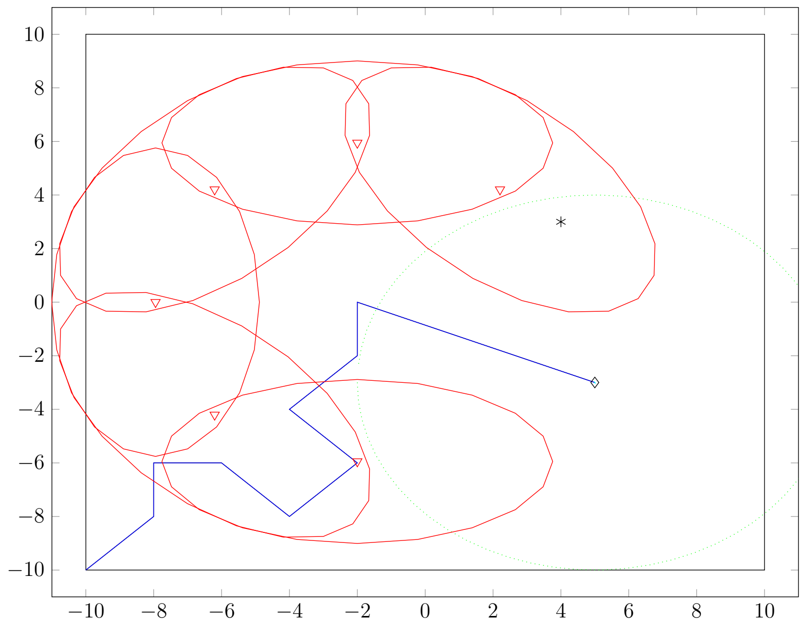


Figure 3.6: Since no measurement is received, the closest hypotheses are down-weighted and pruned.

$$\Pr\left[\text{RECEPTION} \,|\, \hat{d}_{n,h}\right] = \Pr\left[\mathbf{P}_{\mathbf{rx},n} \leq P_{comm} \middle| \, \hat{d}_{n,h}\right] \tag{3.28}$$

$$= \Pr\left[\hat{\mathbf{d}}_n \leq d_{comm} \middle| \, \hat{d}_{n,h}\right] \tag{3.29}$$

$$= \mathbf{\Phi}\left(\frac{\ln d_{th} - \ln \hat{d}_{n,h}}{\frac{\ln 10}{10\eta}\sigma_\Psi}\right) \tag{3.30}$$

where $\mathbf{\Phi}(x)$ is the CDF of the standard normal distribution $\mathbf{Z} = \mathcal{N}(0,1)$. Using this result, the new hypotheses' weights are updated as follows:

$$w_{n,h}^{t+1} = w_{n,h}^{t} \cdot \frac{1 - \Pr[\text{RECEPTION} \,|\, \hat{d}_{n,h}]}{\sum_{h=1}^{H} w_{n,h}^{t} \left(1 - \Pr[\text{RECEPTION} \,|\, \hat{d}_{n,h}]\right)} \tag{3.31}$$

and the transition to log-weights is conducted exactly as (3.25-3.27).

3.4.3 Pruning

To enforce convergence from the GMM to a single Gaussian, some rules to remove useless hypotheses from the filter must be established. Basically, a hypothesis is considered for pruning if it satisfies at least one of the following constraints:

- The associated log-weight $\ln w_{n,h} < 0$ is below a certain threshold. This is the main pruning rule for deleting hypotheses, as the log-weights update processes (3.24) and (3.31) are generally quick to discern unlikely candidates from just a few measurements, or the absence thereof. In fact, the wide shape of the fat-tailed lognormal range distribution employed during the simulated experiments caused some likelihoods to assume very low values, risking a premature convergence with only two or three distance measurements. To avoid the influence of early outliers, the threshold suggested by [37] as $w_{th} = 10^{-5}/H$ was deemed too stringent: a log-threshold of $\ln w_{th} = -(15 + \ln H)$ was found experimentally to work well for the localization problem. The fact that the log-weights are normalized after each update guarantees that the most probable Gaussian has always $\ln w_{n,h} = 0$ and cannot be discarded even with a few outliers.

- The Euclidean distance among two hypotheses that pertains to the same node is smaller than a certain threshold. Since their means can change due to a UKF update, two or more hypotheses can find themselves to be too near to each other due to filter convergence around the true node position As there is no need for such information redundancy, the most unlikely hypothesis (or hypotheses) is removed to save computation time, and the remaining Gaussians have their weights renormalized. Even if [6] suggests setting this threshold to one meter, more often than not this resulted in the filter converging on a bimodal distribution composed of two equally likely Gaussians. These hypotheses would of course be located near one another, but not enough: to discriminate between them, a lot of noisy measurements were needed. A more appropriate choice, verified experimentally, was that of taking the 10% of the biggest length allowed in the deployment area, typically the main diagonal of its bounding box.

This convergence process alone does not always guarantee a correct solution. An unfortunate combination of outliers may lead to the selection of an hypothesis that's far away from the real node position: this error may be exacerbated by incorrect

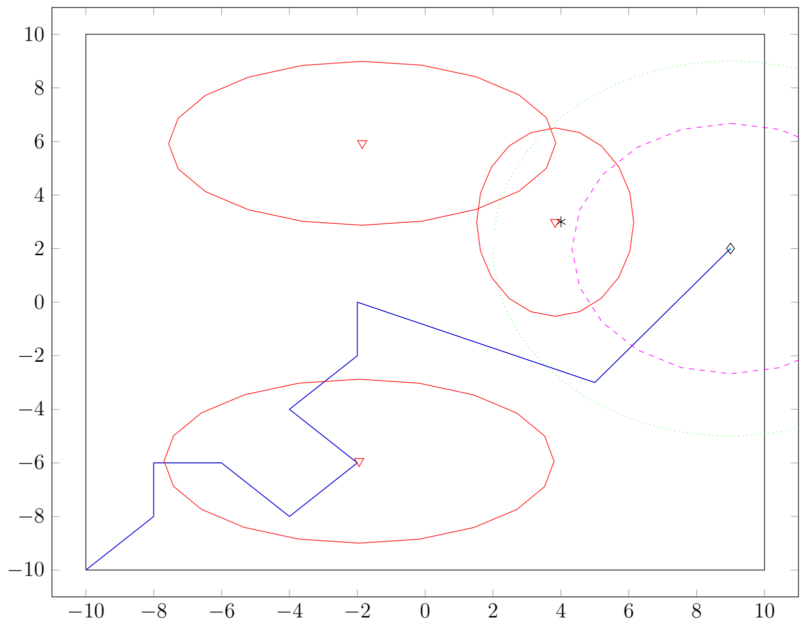


Figure 3.7: Measurement update: the nearest, and thus most likely hypothesis is also the one most influenced by the measurement

assumptions in the filter update. To deal with this kind of critical situations, a system for completely reinitializing a node was implemented.

Quite simply, even when the GMM has collapsed to a single Gaussian for a specific node, the range estimate likelihood is still computed for each new observation, be it failed or not. If the updated weight, renormalized every k measures to maintain a finite memory, is found to be *extremely* unlikely, i.e. if after at most k updates it would already pass the threshold $\ln w_{th}$ defined above, the estimate mean and covariance of the node will be removed from the filter state. Afterwards, a new distance measurement from the same node will be treated exactly as that of a newly discovered node, and a new annulus will be initialized.

During simulation, inherent computational limitations of the Matlab platform caused some distance measurement likelihoods to be equal to 0, and their corresponding logarithms equal to $-\infty$. Such measures would always cause a node reinitialization.

Chapter 4

Motion policies

Mobility, though, can be a double-edged sword. While the advantages of an autonomous MB over static beacon placement has been already discussed, what with its reusable hardware and arbitrary number of measurements, an additional complexity factor is introduced when considering the influence of motion on the robot's battery life. In fact, the AMR's overall energy expenditure during the entire localization process is dominated by the variable cost of moving from a *virtual beacon* site $\mathbf{x}_r(t)$ to the next $\mathbf{x}_r(t+1)$, sometimes even eclipsing the fixed measurement and computational costs that occurs each time interval.

A legitimate question is then raised: is there, and is it possible to find, a MB path over the operative area that can maximize WSN localization accuracy under arbitrary constraints on total time, path length or battery life? Sichitiu [35] first applied the mobile beacon to node localization, and presented the conclusion that the beacon trajectory must cover the entire area in such a way that each node receives at least three non-collinear beacon messages: however, he did not give a specific mobile beacon path. The available literature is still quite scarce on this topic, and can be summarily divided in two main branches: *static* path planning strategies focus primarily on area coverage, disregarding any information acquired by the filter about the actual WSN configuration; whereas *dynamic* path planning strategies make no assumptions about the shape of the operative area, and generally maximize localization by moving the MB as close to the nodes as possible.

Developing a motion policy for RUUMBA can benefit from some of the assumptions made back in Chapter 2. First of all, it should be noted that both time and energy level metrics depend heavily on the total path length of the MB, which can therefore be chosen as the single constraint to impose on the optimization. Then, the undesirable spatial correlation of the shadowing noise $\mathbf{\Psi}$ limits the consistency of measurements taken too close to one another: to maintain unbiased and independent observations, a certain minimum *decorrelation distance* between virtual beacons should be respected. This constraint can also be seen as the minimum length traveled by the MB between discrete time instants: for a typically cluttered indoor environment, its value can be set as low as $d_{decorr} = 1$ m [33].

As a result of that, the operative area lends itself well to a natural grid discretization: starting from the location of the first measure, which usually is the starting position of the MB, a square or hexagonal area tesselation can produce a finite number of virtual beacon *candidate points* from which to observe the WSN. Each of these points would be at least at d_{decorr} from its adjacent neighbors, guaranteeing independence between the respective shadow fadings; moreover, the real advantage would be that of avoiding to evaluate any generic path planning policy on the whole continuous plane. Instead, performance metrics and/or decision criteria would be applied only on the smaller set of candidate points, greatly decreasing computational costs and implementation difficulties.

Even if hexagonal cell shapes would offer a more tight tesselation and an overall increased number of candidate points from which to choose, square tiles were selected for their ease of implementation: when the need for a thorough localization will necessitate a finer grid interval, a practical solution might be to use tex cells [48].

4.1 Formal definition

Formally, a path planning policy is denoted by:

$$\pi \triangleq \langle \pi_0(\mathbf{x}_0), \pi_1(\mathbf{x}_{0:1}), \ldots, \pi_t(\mathbf{x}_{0:t}) \rangle \tag{4.1}$$

where at each step $\pi_t : \mathbf{x}_{0:t} \rightarrow \mathbf{x}_r(t+1)$ the entire state history $\mathbf{x}_{0:t}$ (or better yet, the estimated state history $\hat{\mathbf{x}}_{0:t}$) is mapped to the next target location for the mobile beacon. Since RUUMBA localization is performed with an online Kalman filter, i.e. in an incremental manner, a powerful assumption for this planning problem is that of Markovianity. In other words, the hidden Markov model inherent to Kalman assumptions affirms that the current state estimate at time t contains all of the necessary information to plan the optimal trajectory according to π:

$$\pi_t(\hat{\mathbf{x}}_{0:t}) = \pi_t(\hat{\mathbf{x}}_t) \tag{4.2}$$

This memoryless property finds its usefulness when computing space and time complexities associated with the policy: while computational time still depends on how the path planning strategy constructs its solution, the $O(N')$ memory storage requirements are already satisfied by the current filter state and no additional RAM or disk space is needed.

As a complement to the path planning policy π, an input generating function $\tau : \mathbf{x}_r(t+1), \hat{\mathbf{x}}(t) \rightarrow \Delta_\theta(t), \Delta_D(t)$ can be specifically designed around the motion model to traduce the planar destination selected by π into a series of commands that are understandable by the robot actuators. An additional layer of complexity to the AMR motion, which was not included in the scope of this thesis, can be represented by how the robot navigates when obstacles or non-straight paths are imposed by the environment: in this case, a local pathfinding subalgorithm (for example, A* and its derivatives) may be needed to move between virtual beacon locations.

From an energetic point of view, the expected cost at each time step can be defined in general terms as:

$$C_t = E_M\ (\ \tau(\mathbf{x}_r(t+1), \hat{\mathbf{x}}_r(t))\ ) + \hat{E}_S \tag{4.3}$$

where E_M is the robot actuators' energy consumption for in-place rotations and translations, and $\hat{E}_S$ is the expected energy consumption to collect all available observations at each step. Without loss of generality, the operative area $\mathcal{A}$ will be assumed to be free of obstacles, enabling the robot to always move in a straight line towards its next location. The cost estimate, then, can be approximated by a functional of the Euclidean distance:

$$C_t \approx E'_M(||\mathbf{x}_r(t+1) - \hat{\mathbf{x}}_r(t)||_2) + \hat{E}_S \tag{4.4}$$

Expected sensing costs depends mainly on the expected number of nodes within communication range: without loss of generality, the assumed isotropy of the robot's antenna simplifies the expression of the MB's local neighborhood into that of a circle with radius d_{comm}. Also, where no prior information is available, the WSN is supposed to be uniformly distributed over $\mathcal{A}$, with each of N nodes' position independent from one another. If $\bar{E}_{S,r}$ is the mean energy depleted by the robot for a single RSSI distance measurement, the expected sensing cost for each time step can be estimated as:

$$\hat{E}_S = \Omega\left(\bar{E}_{S,r} \cdot \frac{N}{\mathcal{A}}\mathcal{I}\right) \tag{4.5}$$

where $\mathcal{I} = \mathcal{A} \cap \pi d_{comm}^2$ is the area of the intersection between the operative area and the sensing circle: depending on the context, a conservative approximation of it can be $\mathcal{I} = \min(\mathcal{A}, \pi d_{comm}^2)$. The big omega notation, instead, is because different implementations of the channel access method cannot always be resolved in linear time, as transmission attempts collide more frequently according to local node density: a possible solution is presented in [27]. If it would be desirable to model the *whole* system's energy consumption, accounting also for the nodes' battery depletion, expression (4.5) can be simply modified to add the mean node energy cost:

$$\hat{E}_S = \Omega\left((\bar{E}_{S,r} + \bar{E}_{S,n}) \cdot \frac{N}{\mathcal{A}}\mathcal{I}\right) \tag{4.6}$$

Summing (4.4) along the whole MB trajectory, an incremental expression for the cost function is finally reached:

$$C_{0:T} = E_M\left(\sum_{t=0}^{T} ||\mathbf{x}_r(t+1) - \hat{\mathbf{x}}_r(t)||_2\right) \; + \; (T+1)\hat{E}_S \tag{4.7}$$

This expression alone does not constitute a good performance metric for a localization policy, though: it should be combined with the estimate deviation from the ground truth, evaluated by average error:

$$\mathrm{ME}(\hat{\mathbf{x}}) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \hat{\mathbf{x}}_n||_2 \tag{4.8}$$

or, if a larger outlier influence should be desirable, by *Root Mean Square Error* (RMSE):

$$\mathrm{RMSE}(\hat{\mathbf{x}}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} (||\mathbf{x}_n - \hat{\mathbf{x}}_n||_2)^2} \tag{4.9}$$

In both expressions, it's implicitly assumed that the MB has discovered all of the N nodes in the network, and indeed it is one of the stopping conditions of the filter (see sec. 2.6). However, it could happen that some GMM has not yet reached convergence when the error is computed, leaving more that one Gaussian hypothesis in the state estimate vector $\hat{\mathbf{x}}$, that now is *bigger* than the real state $\mathbf{x}$! In such a case where $N' > N$, a modified metric should be used: to extract the maximum possible amount of information from the GMM, the error for each node is the result of a weighted average between its hypotheses:

$$\mathrm{ME}(\hat{\mathbf{x}}) = \frac{1}{N'}\sum_{n=1}^{N'}\sum_{h=1}^{H} w_{n,h}||\mathbf{x}_n - \hat{\mathbf{x}}_{n,h}||_2 \tag{4.10}$$

$$\mathrm{RMSE}(\hat{\mathbf{x}}) = \sqrt{\frac{1}{N'} \sum_{n=1}^{N'} \sum_{h=1}^{H} w_{n,h} \left(||\mathbf{x}_n - \hat{\mathbf{x}}_{n,h}||_2 \right)^2} \tag{4.11}$$

Formulas (4.10-4.11) and (**??**) can alternately be used as constraint and optimization factors; one can try to obtain the best localization within limited battery life, or spend the least amount of energy to achieve a set confidence about the ground truth.

4.2 Static path planning

The simplest and easiest movement policy is also the most straightforward:

$$\pi_{\mathcal{P}} : \begin{cases} t \to \mathbf{x}_p(t+1) & t = 1, \ldots, T_{\mathcal{P}} \\ t \to \emptyset & t > T_{\mathcal{P}} \end{cases} \tag{4.12}$$

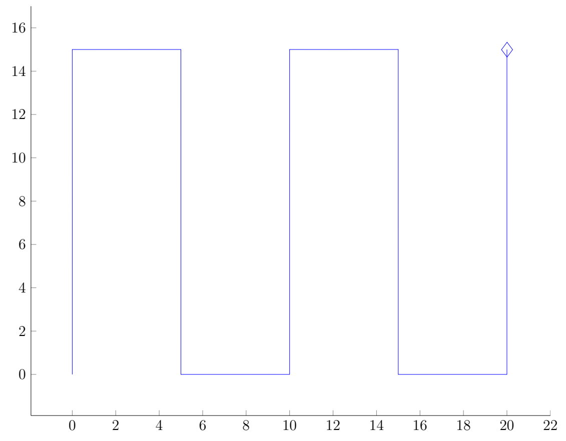
where the ordered set of virtual beacon locations $\mathbf{x}_p(1, \ldots, t+1) \in \mathcal{P}$ belongs to an predetermined path $\mathcal{P}$ composed by $T_{\mathcal{P}}$ planar coordinates. The robot, therefore, follows the given path until no more points are available, whereupon it stops all activities and outputs its current state estimation: this is done regardless of whether the halting conditions of sec. 2.6 are met or not, as it is supposed that all useful observations have already been made. If, however, the metrics on covariance $\mathbf{P}$ suggest that a good enough localization has already been accomplished, there is no need to complete the whole path $\mathcal{P}$ and the localization process can stop.

These kind of paths favor *structure* over *adaptability*: while they're only generally defined for rectangular areas with no obstructions to impede the robot movement, their shape is specifically aimed to optimize geometrical properties like disk coverage of the whole operative area, and collinearities avoidance along the trajectory. Remembering that the lognormal measurement deviation scales with the respective antenna-to-antenna distance, an upper bound on the variance of RSSI ranging can be established by discarding power measures under a certain threshold, thus artificially setting a lower communication radius. Any static path planning policy is then capable to adjust $\mathcal{P}$ such that coverage is still guaranteed, usually at the cost of longer path length.

Some of the most common static paths employed in literature (Scan, Hilbert, k-coverage, random walk) are hereby presented, along with two proposed variants (Squares, Spyral) that find their justification from existing literature.

4.2.1 SCAN, HILBERT

Introduced by Koutsonikolas [21] as a solution to the maximum coverage problem, they are one of the first explicitly defined paths and the *de facto* standard over which to compare eventual alternatives. Scan consists in a simple area traversal along one dimension, as illustrated by Fig. 4.1a: when the pattern followed by the mobile beacon is characterized by a fine enough resolution, the localization error is the lowest among all the alternatives presented in the paper. However, a lot of measurements are taken from points that lie on the same long, straight lines typical of this path: such collinearities can often delay or even impede the resolution of flip ambiguities and, more generally, GMM collapse on a single hypothesis.


(a) Scan path

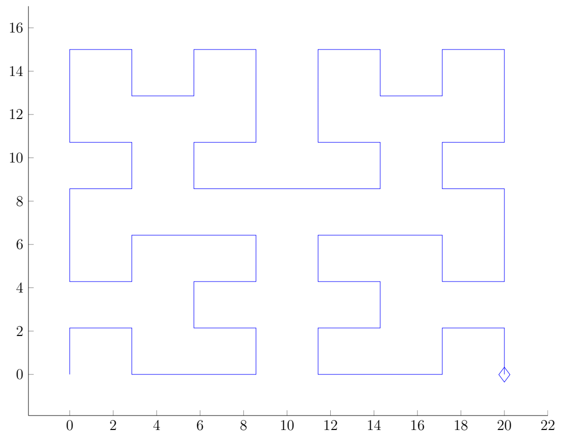

(b) Hilbert path

Figure 4.1

To greatly reduce the number of collinear measurements, the Hilbert space-filling curve was proposed: this pattern creates a linear ordering of points in a higher-dimensional space that can preserve their physical adjacency. A generic n-th order curve divides the bidimensional space into $4n$ square cells and connects the centers of those cells using $4n$ line segments, each with equal length. The reasoning behind the choice of this path is that even though it's longer than Scan and other area coverage patterns, it contains a greater number of turns. Frequent changes of direction localized in a small area lead to precise, noncollinear measurements for the nodes in proximity.

Due to its intuitiveness, SCAN has been one of the most used paths when studying MB-assisted localization: for example, in addition to being already employed in a range-only context [50], it's been also used for range-free algorithms like *Arrival and Departure Overlap* (ADO) [47] and for hybrid algorithms like the *Mobile-assisted RSS and Connectivity* (MRC) localization [41].

4.2.2 k-coverage

Assuming uniform node distribution, Fu et al. [8] chose to approach the problem of optimizing 3-coverage (i.e. aiming to include each point into 3 different communication areas) by partitioning the operative area into hexagonal cell tiles with length proportional to the communicating radius, and to solve a Traveling Salesman Problem among the hexagons' centers. The computation phase is done offline via the *Ant Colony Algorithm* (ACA), which outputs an angular path rich of noncollinearities.

4.2.3 Random walk

When the beacon's antenna can transmit and receive signals from the whole network, that is when the maximum dimension of the operative area is still lesser than the MB communicating radius, Srinath [38] proposed to let the AMR perform a random walk. To avoid the aforementioned shadowing spatial correlation, this should then translate in a random selection between equally-spaced virtual beacon candidate points: this randomized ordering can then be fully performed offline,

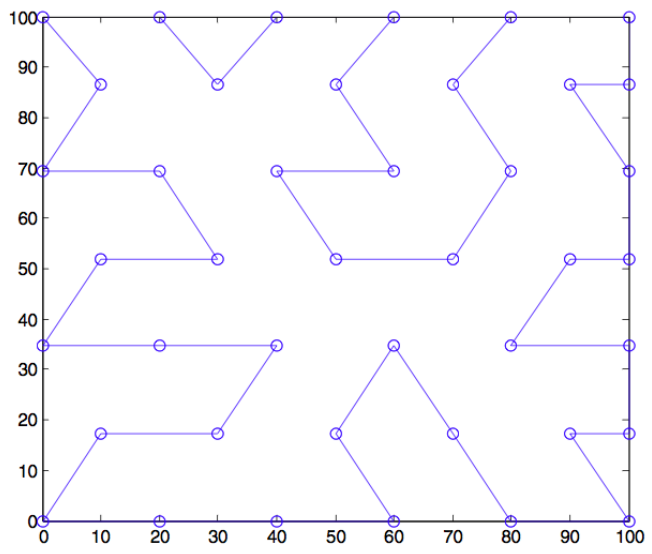


Figure 4.2: k-coverage path

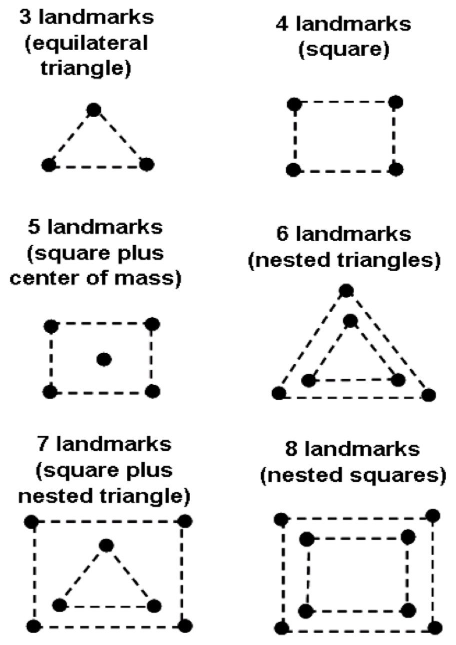
and this policy be equated to a generic static path planning. Relaxing the coverage requirement, this approach works well even when $\mathcal{A}$ is included for the most part, but not totally, in the sensing radius.

4.2.4 SQUARES

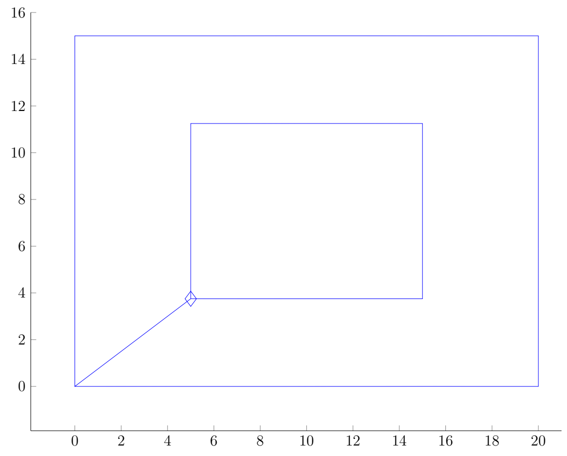
It's common knowledge that good localization results can be obtained when the node is within the convex hull of its neighboring anchors: indeed, centroid [4] and baricentric coordinates [18] methods operate under this assumption. A perimetral disposition of the virtual beacons with respect to the operative area, therefore, seems like a good initial choice: even if the measurements are collinear and the GMMs collapse into a bimodal distribution, one of those two Gaussian hypotheses always falls outside of the operative area and can be immediately discarded.

Of course, innermost nodes still have to reach an acceptable degree of accuracy: this problem can be fixed by moving the MB towards the center, but it's not immediately clear which pattern it should follow. Taking inspiration from the work of Chen et al. [7] about the optimal disposition of a limited number of beacon nodes, it can be seen that the most effective structures are formed by simple shapes enclosed in one another. A concentric square/rectangular structure then emerges:

proceeding along the side of an inner shape, newly discovered nodes should again collapse to two Gaussians on either side of the path. The inward progression, though, allows for a quick deletion of the hypotheses closer to the edges of $\mathcal{A}$; in fact, if a node would had really resided in the "ring" between the last traversed square and the current, it should have already been sensed in the previous iteration, given that the distance between concentric shapes is bounded by the communicating radius.


3 landmarks (equilateral triangle)
4 landmarks (square)
5 landmarks (square plus center of mass)
6 landmarks (nested triangles)
7 landmarks (square plus nested triangle)
8 landmarks (nested squares)

(a) Patterns for optimal landmark deployment



(b) SQUARES path

Figure 4.3

4.2.5 SPYRAL

Originally developed as an octagonal variant of SQUARES, this pattern bears also some resemblance to the CIRCLES configuration proposed by Huang in [12], and can be seen as a hybrid between the two that occupies a middle point between the respective advantages and disadvantages. In fact, the octagons perform better when considering corner coverage, one of the weak points of a circular shape, and their overall path length is slightly shorter than square trajectories, because corners are literally cut. As a last point, approximating curvilinear paths with straight lines should be seen as desirable when recalling the robot motion model (2.5): the long-term influence of rotational noise ν_θ on the actual robot path cannot be discounted, and hence the least amount of in-place turns should be planned.

A similar pattern was used as MB trajectory by Sun in [40]; in that paper, the only analysis performed about the path influence on localization was to vary the curvature degree of the helix shape, showing a steep rise of RMSE on a range from $\kappa = 1$ to $\kappa = 2$.

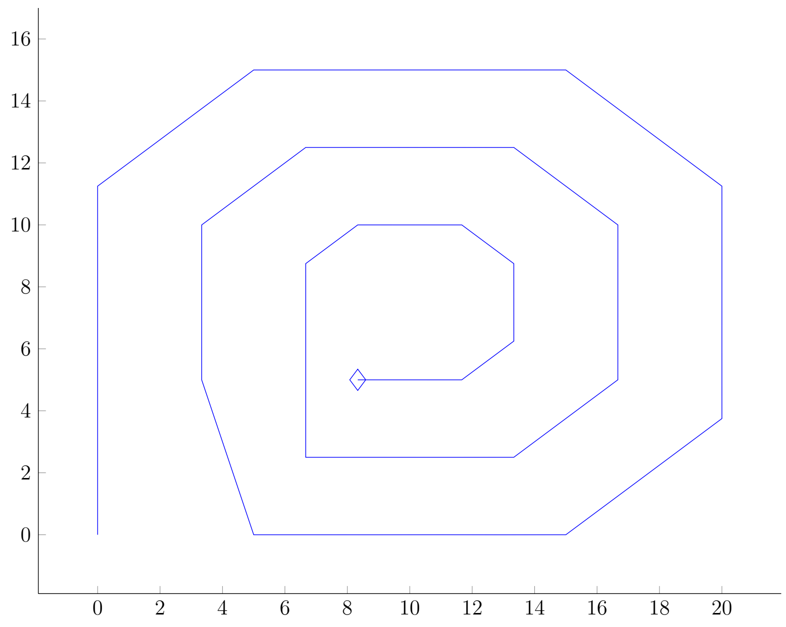


Figure 4.4: SPYRAL path

4.3 Dynamic path planning

It's not always the case that a realistic setting can provide enough map information about the operative area, though. Considering the example of indoor localization, most static paths cannot be applied to complex floorplans without first dividing them into adjacent convex shapes: in other cases, node distribution may not be uniform over $\mathcal{A}$ and instead the WSN may have higher density around a small area of interest. It might be more practical, then, to define a movement policy capable of adapting the AMR's path to the specific context over which it's deployed:

with almost no exception, this idea translates into using the filter state estimate $\hat{\mathbf{x}}$ and/or the observations $\mathbf{z}_t$ (usually acquired in a short time interval) to move towards the location with least expected measurement variance.

These kind of policies are not without shortcomings: the main one being that they must necessarily be computed online, because the optimal path could change whenever new data is acquired. Remembering that these recurrent computational costs are imposed on the limited hardware resources of the MB, there's a very high risk that path planning can bottleneck the overall localization execution time: particular care, then, has to be given when evaluating the efficiency and complexity of each policy.

Because of that, most strategies avoid the general case of planning a complete *most informative path* for the whole network, spanning the continuous spaces of state and action; instead they assume relaxed constraints such as the already mentioned grid and time discretizations, accurate knowledge of the robot location, and the adoption of *myopic planning*.

This latter assumption, which can be interpreted as imposing an upper bound on the number of steps to plan in advance, is crucial: indeed, each dynamic $\pi(t)$ contains some sort of optimization of the predicted evolution $\hat{\mathbf{x}}(t+1)$ of the high-dimensional state vector $\mathbf{x}(t)$, which itself is a function of the robot motion decided by $\pi(t-1)$. The overall complexity for a planning horizon composed by T steps is hence exponential in T: moreover, the high state variability due to both node discovery and localization refinement greatly devalues the optimality of any path based on information that is more than a few steps old, forcing its periodic recomputation.

For these reasons, to the knowledge of the author no practical planning strategies have been proposed with a planning horizon greater than $T = 3$ steps; to avoid exponential terms altogether, many policies instead make recourse to sub-optimal greedy schemes and plan only the next step at each iteration.

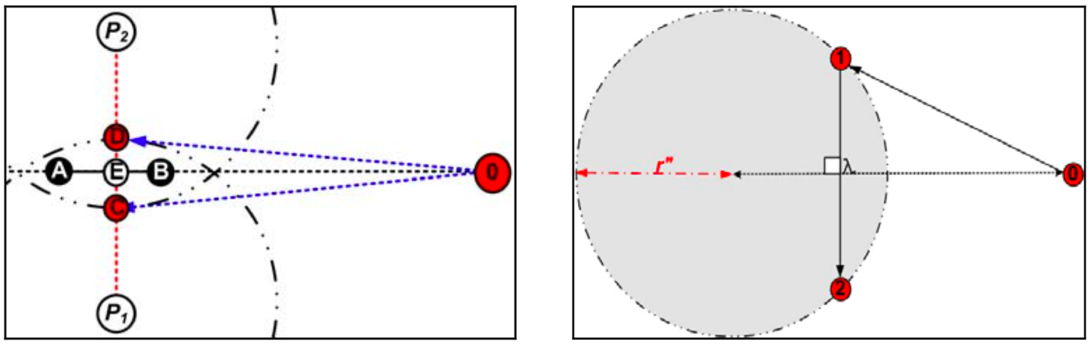
Some of the most relevant dynamic path planning policies present in literature (MBAL, DREAMS, PCRB minimization) are hereby presented and summarily described; in addition to that, a collection of four greedy policies are proposed. Two of them (Greedy-loc, Greedy-gmm-loc) consist in a very simple ranking that can be done in linear time, while the third (Greedy-P) finds the maximum of a multimodal Gaussian; the last proposal (SIGH) takes inspiration from the field of information theory and provides a heuristic for maximizing the information gain of a measure.

4.3.1 MBAL

The *Mobile Beacon-Assisted Localization* algorithm was created by Kim and Lee [20] for 2-D localization and successively adapted for the tridimensional case with an aerial mobile beacon [19]. It begins by moving to the middle of the operative area and performing a reference movement with triangular shape; the nodes within the triangle hull can trilaterate then themselves, and serve as additional beacons to assist their neighborhoods.

If the network possesses certain topological properties (3-connectivity, clique rigidity) that are often related to node density, then this process can be recursively repeated by the fringe nodes. At each iteration, unlocalized nodes that receive three noncollinear beacon signals can compute their respective position and change their status as beacons for the next step, until a complete multi-hop network self-localization is achieved. Unfortunately, nodes situated at the frontier of the operational area or over sparse areas often suffer from low connectivity: they then turn themselves into *Request Nodes*, and broadcast their neighborhood size by transmitting their status as “Two-RN”, “One-RN” or “Zero-RN” along with their partial location information.

When the MB receives their message, it can deduct the optimal area where a measurement would improve upon the geometrical constraints: for Two-RN it’s



(a) Two-RN node: MB elects to move to position C or D

(b) One-RN node: MB selects two extremes of a chord with minimum length λ

Figure 4.5: Handling of unsufficiently constrained nodes by MBAL.

the area intersection between the two symmetrical hypotheses, while for One-RN it's two point on a long enough chord around the only neighboring beacon. Due to their large candidate area, Zero-RN nodes are not handled until the propagation effect described above localizes one of their neighbors, turning them into One-RN. Finally, the path is selected greedily: at each time step, the MB moves towards the closest candidate area.

4.3.2 DREAMS

The acronym stands for *DeteRministic bEAcon Mobility Scheduling*: while assuming no knowledge about the operative area boundaries, this algorithm by Li et al. [24]is deterministic in the sense that sensors' visiting order is fixed provided that the MB starts from the same position.

Exploiting trigonometric computations, the devised pattern can unambiguously discover the direction of a nearby node even when local noise distorts the communicating area: after the MB has moved arbitrarily close to the node, localization is achieved using the AMR's position. The robot then proceeds to the next target, performing a Depth-First traversal of the whole network graph; this path can be further shortened by eliminating nodes already localized via previous measurements, and by constructing Local Minimum Spanning Trees that can approximately solve

Euclidean Traveling Salesman sub-problems.

Fig. **??** shows the mobile beacon, originally in position p, while trying to get close to node S: intermediary positions $q_{1,\ldots,4}$ represent a complete (albeit failed) iteration of the algorithm, that starts by moving randomly to q_1 but ultimately is forced to default to the starting position as no progress is made on steps 2 to 4. By contrast, measurements performed on positions $q_{5,6,7}$ notice a RSSI increase and move accordingly.

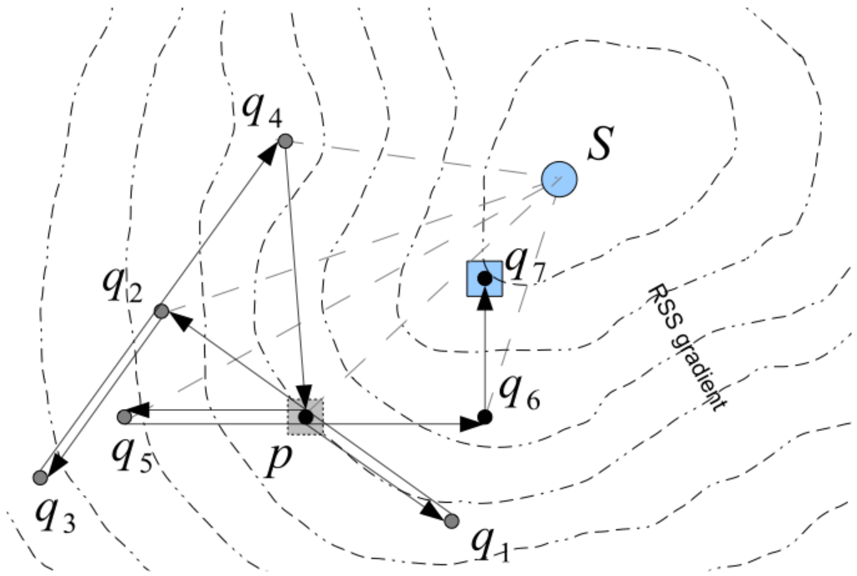


Figure 4.6: DREAMS exploratory pattern for noisy measurements.

4.3.3 PCRB minimization

The scheme proposed by Martinez [?] does not actually plan a specific path, as much as it builds a reinforcement learning strategy to find that movement policy: the additional assumption of a rough starting initialization can be met trough network self-localization. Representing the policy with a parametrized form $\pi(\mathbf{\Theta})$, a Bayesian regression constructed via Gaussian Processes can map the parameters $\mathbf{\Theta}$ to the cost function; sampling only the parameter values that correspond to a high GP variance or to a low expected cost corresponds to choosing between exploration and exploitation behaviors. To choose the sampling locations, the concept of infill functions is borrowed from the geostatistics literature.

Such a general solution, quite honestly, feels excessively cumbersome when applied to the restricted application of WSN localization. Still, some of the accompanying results are remarkable, first of all the choice of approximating the cost function with the expected *Posterior Cramèr-Rao Bound* (PCRB) for nonlinear systems: this alternative is certainly cheaper than predicting the posterior state estimate for every choice of $\mathbf{x}_r(t+1)$, and is moreover agnostic to the type of localization filter. It is defined as the inverse of the Fisher information matrix $\mathbf{J}$ and serves as an upper bound on the maximum information that can be extracted from the system with a given measurement model.

Unluckily, the computation of this bound is complicated by the nonadditiveness of the noise factors: in the same publication, a comparison between two different methods for approximating the true PCRB is made. A tight bound on the cost can be assessed by employing jump Markov linear models, but at the price of very high computational costs; alternatively, forcing the use of an additive noise representation, a single Riccati-like recursive equation can be cheaply solved to obtain a much looser bound.

Looking back at the case of WSN localization, its inherent discretizations (especially if limiting the space of actions to move only through adjacent cells) could cut back on the computational costs, rendering this metric usable in real-time applications; otherwise, the ordering induced by a loose bound between candidate points could still be used as an approximation of the true ranking. Due to time constraints, these claims could not be further investigated.

4.3.4 Greedy

These new proposed methods all share some common properties that makes them alike: they are all based only on the state estimate $\hat{\mathbf{x}}$ and covariance $\mathbf{P}$, they all count only those nodes and/or hypotheses that are not yet localized according to the formulas in section 2.6, and they all default to a random selection among candidate points whenever the state vector does not contain any unlocalized node.

Being greedy, all these policies have a receding horizon length of 1, that is they plan only the next step: their simplicity makes them lightweight enough to be computed in linear time, with $O(N')$ operations.

Greedy-loc

This strategy continuously moves toward the most precise node position estimate, selected from those whose eigenvalues are still above the set threshold for filter stopping. Effectively, it tries to quickly localize one node at a time to remove it from the vector and proceed to the next: by focusing on those nodes that are closer to completion, the MB can meanwhile collect more observations on the others.

Given a discretization $\mathcal{D}(\mathcal{A}) = \mathbf{x}_d^1, \mathbf{x}_d^2, \ldots$ of the operative area $\mathcal{A}$, the point selected as next beacon location will be the closest to the "best" state position estimate, measured according to a ranking between covariances:

$$\pi_{loc} : \mathcal{D}, \hat{\mathbf{x}}, \mathbf{P} \rightarrow \arg \min_{\mathbf{x}_d^i \in \mathcal{D}} ||\mathbf{x}_d^i - \hat{\mathbf{x}}[\bar{n}]||_2 \tag{4.13}$$

with

$$\bar{n} = \arg \max_{n=1 \ldots N'} \frac{1}{\det \mathbf{P}[n]} \tag{4.14}$$

and $\hat{\mathbf{x}}[i]$ and $\mathbf{P}[i]$ indicating the estimate and 2×2 square covariance matrix that pertain to the i-th node. Due to logarithm monotonicity, this formulation is equivalent to a ranking on the *self-information* $I = \log(1/\mathbf{P})$ of each estimate:

$$\arg\max_{n=1\dots N'} \frac{1}{\det \mathbf{P}[n]} = \arg\max_{n=1\dots N'} I(n) = \arg\min_{n=1\dots N'} \log\left(\det \mathbf{P}[n]\right) \tag{4.15}$$

Greedy-GMM-loc

Seeing that a high number of hypotheses in the state estimate is undesirable, as it needlessly slows down computation, the above policy can be easily modified to improve upon the speed of GMM convergence. When the filter contains uncollapsed nodes, a good way of rapidly alter the weight of any single Gaussian h is to perform as little as a single measurement while being close to its mean: if the node is nearby, the associated weight will greatly increase at the expense of every other hypotheses; otherwise, that estimate is flatly wrong and $w_{n,h}$ will greatly decrease, eventually triggering a removal from the state vector.

To remove the most number of hypotheses in a short amount of time, it suffices to recall their starting annular configuration: specifically, Gaussians that are close, with respect to the angular coordinate ϕ, to an incorrect hypothesis are more probable to be incorrect too, and consequently to have similar weights. This policy, then, makes the MB always move towards the least precise but most probable hypothesis of a GMM, reasoning that any measurements performed in that area will either improve localization or lead to a quick pruning. In this way, the contribution of each observation is not "wasted" by focusing on refining an already precise node position or on deleting an already improbable hypothesis.

Translating these concepts into formulas, they correspond again to selecting the candidate point $\mathbf{x}_d^i$ which is closest to the desired location:

$$\pi_{GMMloc} : \mathcal{D}, \hat{\mathbf{x}}, \mathbf{P} \rightarrow \arg\min_{\mathbf{x}_d^i \in \mathcal{D}} ||\mathbf{x}_d^i - \hat{\mathbf{x}}[\bar{n}]||_2 \tag{4.16}$$

where the target is now found by ranking on both the weight within a GMM and the hypothesis' covariance:

$$\bar{n} = \begin{cases} \arg \max\limits_{n=1 \ldots N'} w_{n,h} \det \mathbf{P}[n,h] & \text{if } \exists \, GMM \in \hat{\mathbf{x}} \\ \arg \max\limits_{n=1 \ldots N'} \dfrac{1}{\det \mathbf{P}[n]} & \text{if } \nexists \, GMM \in \hat{\mathbf{x}} \end{cases} \tag{4.17}$$

By process of elimination, all that will eventually be left is a state vector containing only the best hypotheses: in that case, the policy will default to Greedy-loc, trying to improve each node's precision until localization is complete.

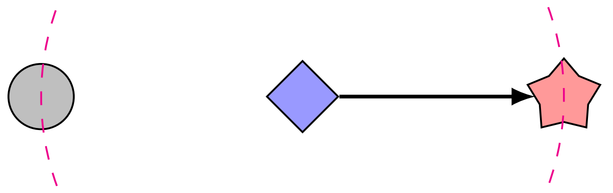
Overshooting

While testing the previous two policies, simulations highlighted a counterintuitive result: namely, that performing multiple measurements while moving from a target location to the next would often *worsen* the overall WSN localization accuracy and speed, despite collecting more data about the nodes' true position. For the specific cases of Greedy-loc and Greedy-gmm-loc, it happened more than a few times that the mean estimate of the most valuable Gaussian would be "chased" by the MB, constantly moving away from it until colliding with the operative area bounds.

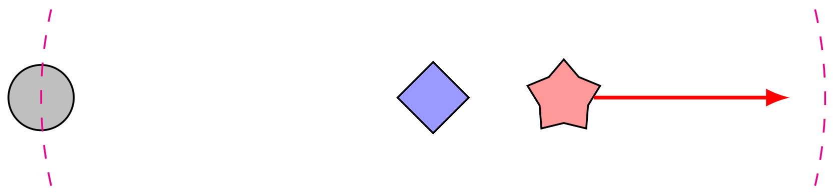
This phenomenon can be explained by looking at the one-dimensional case of Fig. 4.7: if, for some unspecified reason, the (blue) mobile beacon finds itself between its target (red) state estimate and the associated (grey) true node position, according to its policy it will try to move toward the former. If, along this path, it makes a stop to collect a new measurement, a longer distance will be perceived: the online filter, then, will update the red hypothesis along the direction of minimum effort, that is directly away from the MB. This event is then repeated as long as that position estimate is the one targeted by the robot.

The obvious solution would be to defer any measurement up until the target location is reached: still, it should be recalled that the robot does not move to the precise mean of its intended Gaussian, but to the closest point in the area discretization. If that point is between node and hypothesis, the problem can reappear: to guarantee that this doesn't happen, the concept of *overshooting* is introduced. Simply put, when τ would plan the control inputs to move the robot, an additional fixed term would be added to Δ_D: this quantity should be greater or equal than the distance between candidate points, so that both node and estimate stay on the same side of the MB.

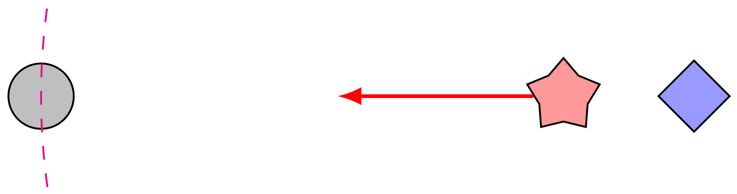
All the previous reasoning is still valid when considering planar configurations: in fact, to know if and how much this problem would present itself, it's sufficient to project the robot's position on the line connecting a node with its mean estimate, and afterwards interpret it as the 1-D case.



(a) The MB plans its movement



(b) With intermediate measurements



(c) With overshooting

Figure 4.7: Chasing and overshooting effects for one-dimensional localization.

Greedy-P

The basic intuition behind this last movement policy is that it's not really important choosing *which* node is localized, as long as the whole network is. By combining all the unimodal Gaussian estimates/hypotheses into a multimodal mixture distribution

f defined over the whole operative area and evaluated on the discretization $\mathcal{D}$, it's possible to build a Maximum Likelihood estimator for the virtual beacon candidate with least expected distance to any one node.

$$\pi_P : \mathcal{D}, \hat{\mathbf{x}}, \mathbf{P} \rightarrow \arg\max_{\mathbf{x}_d^i \in \mathcal{D}} f(\mathbf{x}_d^i) \tag{4.18}$$

$$f(\mathbf{x}) = \sum_{n=1}^{N'} \sum_{h=1}^{H} w_{n,h} \cdot \mathcal{N}(\hat{\mathbf{x}}_{n,h}, \mathbf{P}[n,h]) \tag{4.19}$$

It could happen that for particular symmetric configurations of the wireless network, the measurement performed at the coordinates selected by π_P do not carry enough information to sensibly influence the maximum of f: this could happen when no significative localization improvement, node completion or hypothesis removal is possible. The policy would then be stuck, always selecting the same point from the discretization: this eventuality is avoided by selecting without replacement.

4.3.5 SIGH minimization

To reduce the number of measurements, and hence indirectly the length of the mobile beacon path, one should aim to maximize the significance of each RSSI range observation. This concept can be represented by *information gain*,[1] a way of measuring the potential contribution to overall localization.

In Bayesian statistics and information theory, this quantity is defined as the expected decrease of Shannon differential entropy from a prior to a posterior distribution; within the limited scope of localizing a single node n, active sensing policies [39] [30] [5] try to find the maximal expected information gain for each possible robot location $\mathbf{x}_r$:

$$G_n(\mathbf{x}_r) = \mathrm{E}\left[I(\hat{\mathbf{x}}_n, \mathbf{z}|\mathbf{x}_r)\right] = H(\hat{\mathbf{x}}_n) - \mathrm{E}_{\mathbf{z}_{t+1}}\left[H(\hat{\mathbf{x}}_n|\mathbf{z}_{t+1}, \mathbf{x}_r)\right] \tag{4.20}$$

[1] Also known as Kullback-Leibler divergence, or mutual information.

where the expectation is defined over all possible measurements. This computation is often quite onerous: even with good knowledge about both measurement model and channel parameters, analytical solutions are infeasible, and discretized approximations have cubic complexity. Moreover, the chasing phenomena discussed above prevent the use of a commonly implemented simplification, namely that of constraining the selection of $\mathbf{x}_r$ to move only trough adjacent cells; finding the best MB position among K points in the discretized area, then, is $O(Km^3)$.

An alternative approach was theorized, and successively validated by existing literature [44] [25]: given that mutual information $I(\hat{\mathbf{x}}_n, \mathbf{z})$ is symmetric, a way to reduce the dimensionality of equation (4.20) is to rewrite it by switching the conditioned variables. The formula now becomes:

$$G_n(\mathbf{x}_r) = \mathrm{E}_{\hat{\mathbf{x}}_n}\left[H(\mathbf{z}_{t+1}|\mathbf{x}_r)\right] - H(\mathbf{z}_{t+1}|\hat{\mathbf{x}}_n, \mathbf{x}_r) \tag{4.21}$$

which can be interpreted as the difference between total measurement uncertainty and the specific contribution due to modelization/noise. This change alone does not help with complexity, as $p(\mathbf{z}|\mathbf{x}_r)$ must still be computed for every configuration on the state space. It should be noted, though, that $\mathbf{z}$ is a noisy lognormal observation of the sufficient statistic $\mathbf{v} = ||\hat{\mathbf{x}} - \mathbf{x}_r||_2$, i.e. the projection of a two-dimensional[2] location parameter on a one-dimensional sensor observation perspective. Specifically, this *view distribution* models the distance between $\mathbf{x}_r$ and the node position estimate itself:

$$p(z|\mathbf{x}_r) = \int_v p(z|v)p(v|\mathbf{x}_r)\mathrm{d}v \tag{4.22}$$

$$p(v|\mathbf{x}_r)\mathrm{d}v = \int_{v \le ||\mathbf{x}_r - \hat{\mathbf{x}}_n||_2 \le v+\mathrm{d}v} p(\hat{\mathbf{x}}_n)\mathrm{d}\hat{\mathbf{x}}_n \tag{4.23}$$

The whole distribution can be computed in quadratic time with an m-binned histogram approximation of the integral in 4.23; that, in turn, is obtained by sampling the Gaussian distribution $\hat{\mathbf{x}}_n$ with a grid of $m \times m$ points.

[2] Four-dimensional, if the robot's position is uncertain too.

Wang and Yao [44] suggested that, under the assumption that there exist some potential location $\mathbf{x}_r$ where the measurements are sensibly more informative than the average, the ranking induced by G_n does not significantly differ from the one based on the following entropy difference:

$$
\begin{aligned}
G_n(\mathbf{x}_r) &= \mathrm{E}_{\mathbf{v}}\left[H(\mathbf{z}_{t+1})\right] - H(\mathbf{z}_{t+1}|\mathbf{v}_n) && (4.24)\\
&\approx H(\mathbf{v}_n) - H(\mathbf{z}_{t+1}|\hat{\mathbf{v}}_n) && (4.25)
\end{aligned}
$$

The two terms in the last equation are the view entropy, which can be regarded as a noise-free measurement entropy where $p(z|v)$ is assumed to be deterministic without uncertainty; and the sensing entropy, computed using the Maximum Likelihood estimate for the node position:

$$
H(\mathbf{z}_{t+1}|\hat{\mathbf{v}}_n) = \frac{1}{2} + \frac{1}{2}\ln(2\pi\sigma^2) + \ln||\hat{\mathbf{x}}_n - \mathbf{x}_r||_2 \tag{4.26}
$$

It's now possible to define the information gain with respect to the whole network, including the weighted contribution of uncollapsed GMMs:

$$
G(\mathbf{x}_r) = \sum_{n=1}^{N'}\sum_{h=1}^{H} w_{n,h} G_{n,h}(\mathbf{x}_r) \tag{4.27}
$$

To find the optimal candidate point $\mathbf{x}_r$ over which to move the robot, both its associated information gain and the expected distance needed to reach it it are needed: as they have to be compared to each other they are normalized on the interval $[0, 1]$:

$$
\overline{G}(\mathbf{x}_r) = \frac{G(\mathbf{x}_r) - \min_{\mathbf{x}_r} G(\mathbf{x}_r)}{\max_{\mathbf{x}_r} G(\mathbf{x}_r)} \qquad \overline{D}(\mathbf{x}_r) = \frac{D(\mathbf{x}_r) - \min_{\mathbf{x}_r} D(\mathbf{x}_r)}{\max_{\mathbf{x}_r} D(\mathbf{x}_r)} \tag{4.28}
$$

The final policy for maximizing this *Simple Information Gain Heuristic* (SIGH) metric will use a tradeoff parameter α to fine-tune the cost function:

$$
\pi_{SIGH} : \mathcal{D}, \hat{\mathbf{x}}, \alpha \rightarrow \arg\max_{\mathbf{x}_r \in \mathcal{D}} \alpha \cdot \overline{G}(\mathbf{x}_r) + (1-\alpha) \cdot \overline{D}(\mathbf{x}_r) \tag{4.29}
$$

Chapter 5

Simulation

BEFORE choosing one movement policy above the others, a standardized comparison should be done, either by means of a computer simulation or by physical experimentation on a real test case. Regrettably, available literature cannot agree on a single WSN configuration over which to analyze the respective performances: without the time to adapt each and every policy to a single framework, the schemes proposed in this thesis will be measured against only to static paths. To compare them to other dynamic policies, the author refers to the cited sources and trusts the reader's discernment.

A foremost difference between RUUMBA and most MB-assisted localization techniques is that the memoryless property is fully enforced: past measurements are not stored until completion, ready to be computed all at once, but instead they are integrated in the state vector at each time step. Usually this should not be a cause of problem, as Kalman filters are particularly efficient for handling streams of observations. Still, geometrical constraints for localizability suggest that optimal measurements should be performed according to certain structures: without keeping track of the full AMR's path, no assessment of clique rigidity or noncollinearity can be done. The value of these assertions is even greater in the presence of measurement noise, as the feedback introduced by outliers on policies' choices can cause filter divergence before being averaged out.

All simulations were conducted using Matlab R2010a on a MacBook with 2.4 GHz dual core processor: the simulating software has been written to be as modular as possible, allowing future implementations to build upon it. Motion and measurement models, node initialization and movement policies can be substituted to the proposed algorithms to perform comparisons that are as standardized as possible. All code has been fully commented and documented.

5.1 Layout

Because of computational reasons, the test network was restricted to only 6 nodes, randomly deployed on a rectangular operative area having sides 20 m and 15 m long. To provide consistency with the static paths, the starting point of the mobile beacon trajectory has been fixed at the lowest left corner; the area has been discretized into 350 400 hexagonal cells, each with distance of 1 m from its neighbors, and their centers were set to be the candidate points for virtual beacon deployment.

Environmental and noise parameters were set trying to adhere to realistic conditions that can be found in a typical office space: specifically, path loss exponent was set to $\eta = 2$ and shadowing noise deviation was chosen to be $\sigma_\Psi = 3$. This latter value assumes multichannel averaging, though, and may be considerably higher whenever this condition is not satisfied.

Static policies perform their measurements every 2 m or whenever the path does a turn; random and dynamic policies only do so when arriving at their desired locations. The SIGH movement policy, finally, has additionally been tested with differing proportions of how information and distance influence the decision of a target position. Values of $\alpha = 1.0$, 0.9, 0.8, 0.7, 0.6, and 0.5 have been experimented upon, always favoring informativeness to proximity.

Node initialization consists in $H = 8$ hypotheses, which are pruned if their log-weights fall below e^{-15}/H or if they are closer than 2.5 m from each other. A procedure was added, such that every hypothesis that would exit the area is scaled back into it, keeping its variance but shifting the mean to the intersection point between the boundary and the line going from robot to the position mean.

The whole simulation consisted in 300 consecutive runs under the same conditions: metrics foe computation time, accuracy, precision, number of observations and path length were averaged and compared.

5.2 Results

Static paths, simply put, outperform dynamic policies. Some of the motivations have already been discussed: summarizing them, the focus on area coverage and non-collinearity shared by most static policies imparts a structuralness property to the process of data acquisition that a dynamic path cannot provide. Since the cost of measurements is low, there is no need to immediately move to the most informative point, wasting energy on long paths and doing a lot of observations in a tight cluster. Instead, the coverage constraint causes the measurements to be better spread and diversified over the whole area.

The intuition behind SQUARES and SPYRAL proved to be a good one, as they outperform classic SCAN patterns with just a little more traveled distance: enclosing a node within concentric convex hulls significantly helps with its localization. The octagonal shape performs slightly better, probably in virtue of its higher number of noncollinearities: moreover, it's the most consistent path as it has the lowest error deviation.

Between the dynamic paths, GREEDY-P and GREEDY-LOC score similarly poorly, maintaining a low path length but missing the nodes' true position by almost 3 m: contrariwise, the accurate localization obtained by random selection among candidate points is opposed by huge motion costs. The best alternative seems to use Greedy-gmm-loc, as the paths linking the least localized hypotheses are often more spatially spread and offer more coverage. Finally, overshooting is proven to be a favorable option, even if it adds to the overall traveled distance.

Information-based policies are average at best, even ignoring motion costs: moreover, electing to move along shorter paths uniformly worsens the final localization error. If the robot's behavior while using the SIGH minimization policy is observed, inefficient decisions can be noted: the foremost of them is the constant back-and-forth that occurs when a GMM is reduced to two specular hypotheses with similar weight. In that case, the MB tries to resolve the ambiguity by moving towards one Gaussian, which lowers its weight and increases the other's; this change now makes the algorithm be attracted to the opposite hypothesis, and the cycle repeats. Eventually, there are no more candidate points around the two still uncollapsed terms of the GMM, and the metric decides that the best course of action is to observe them from afar to slowly reach convergence.

Accompanying mediocre results, the computational time to execute these policies is an order of magnitude greater than their greedy and static competitors: to fully localize 6 nodes in a small area from start to stop, more than 30 s are needed. It's the author's conclusion, then, that this kind of policy is not cost-effective in any sense, and shouldn't be pursued.

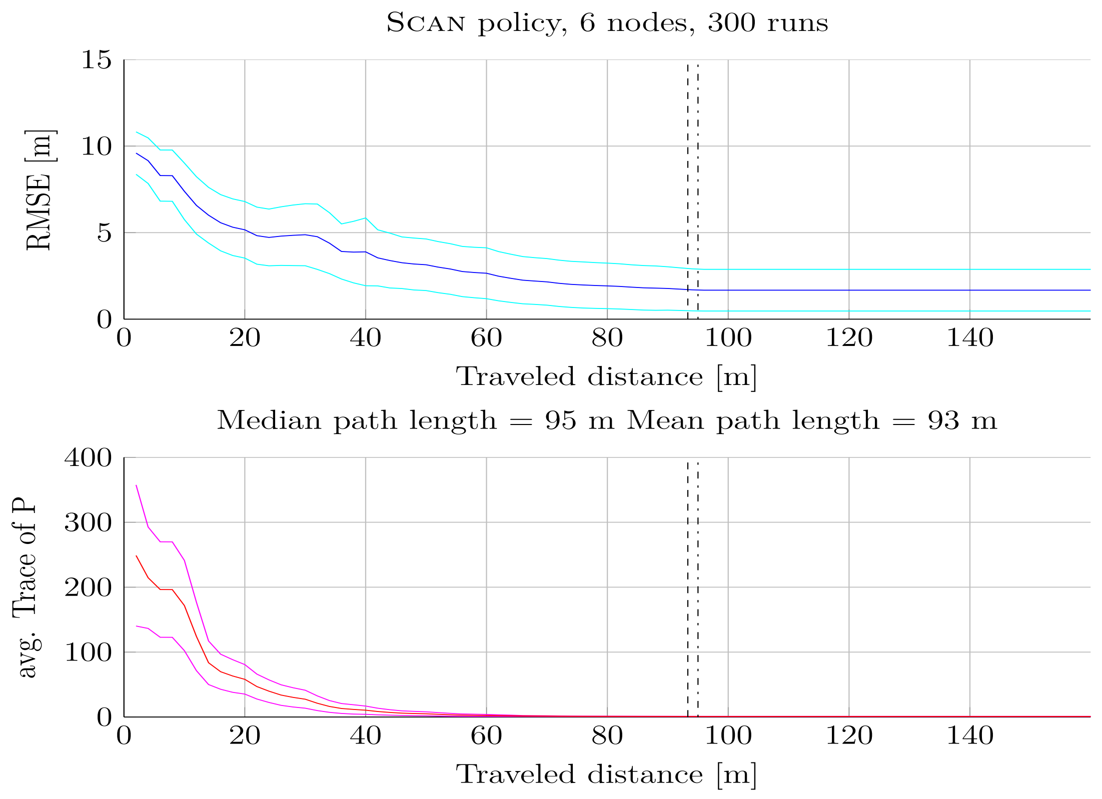
The following figures plot the evolution, for each policy, of RMSE (blue line) and trace of covariance (red line) as functions of total path length. A confidence interval of one standard deviation is drawn around each plot; where present, a vertical dashed line and a vertical dot-dashed line represent respectively the mean and median completion time for the localization procedure. All plots have been scaled to a single reference frame, as to facilitate comparisons between policies.

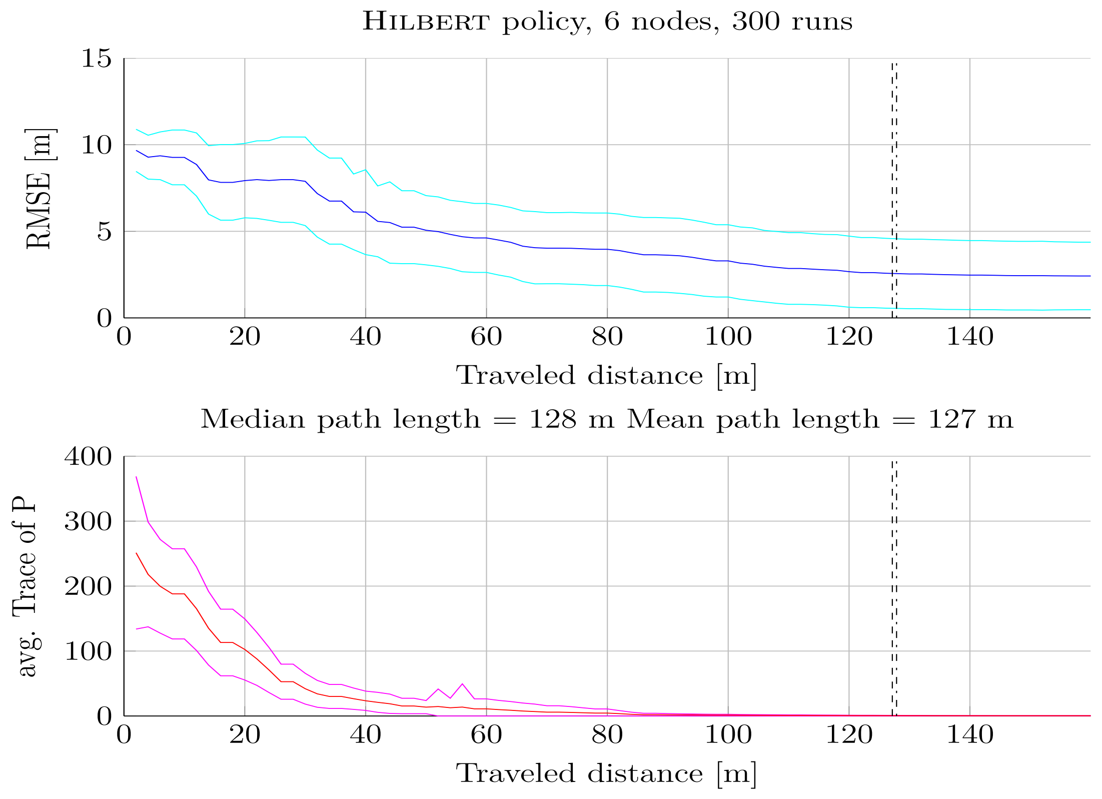
For some of these policies, the trace does not monotonically decrease but presents little bumps: they are the result of a re-initialization of some node, which happens when some estimate is grossly off target and the associated measurement likelihoods are abysmaly low.

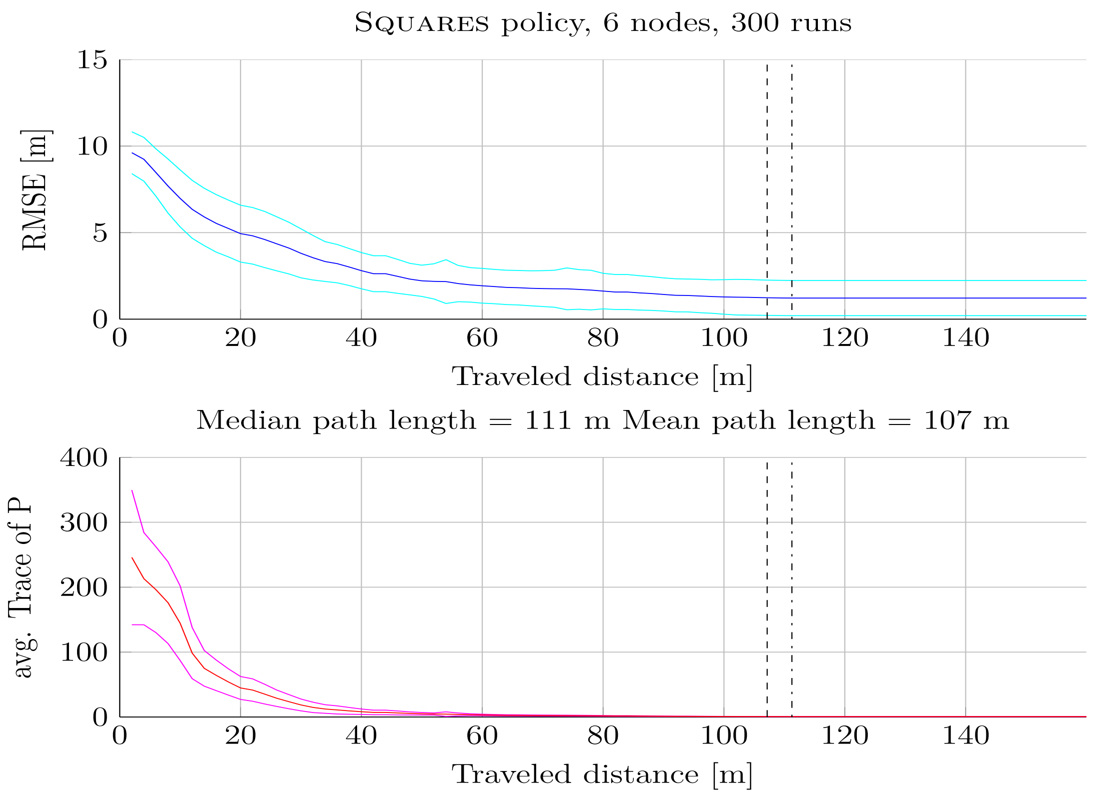
Policy	ME	RMSE	Path length	# obs.
Scan	1.26 m	1.68 m	93.32 m	39.3
Hilbert	1.62 m	2.42 m	127.16 m	53.1
Squares	0.96 m	1.22 m	107.15 m	50.2
Spyral	0.89 m	1.08 m	106.72 m	47.9
Random walk	1.04 m	1.50 m	366.75 m	62.1
Greedy-loc	2.91 m	4.26 m	113.81 m	41.2
Greedy-loc-o	2.16 m	3.28 m	144.55 m	41.8
Greedy-gmm-loc	1.72 m	2.51 m	125.16 m	31.8
Greedy-gmm-loc-o	1.24 m	1.76 m	157.80 m	35.6
Greedy-P	2.96 m	4.42 m	128.69 m	35.7
SIGH, $\alpha = 1$	1.71 m	2.55 m	194.64 m	36.7
SIGH, $\alpha = 0.9$	1.79 m	2.65 m	175.75 m	38.0
SIGH, $\alpha = 0.8$	1.89 m	2.83 m	151.24 m	38.1
SIGH, $\alpha = 0.7$	1.92 m	2.88 m	134.83 m	38.6
SIGH, $\alpha = 0.6$	2.19 m	3.37 m	113.50 m	38.6
SIGH, $\alpha = 0.5$	2.42 m	3.78 m	98.56 m	41.3

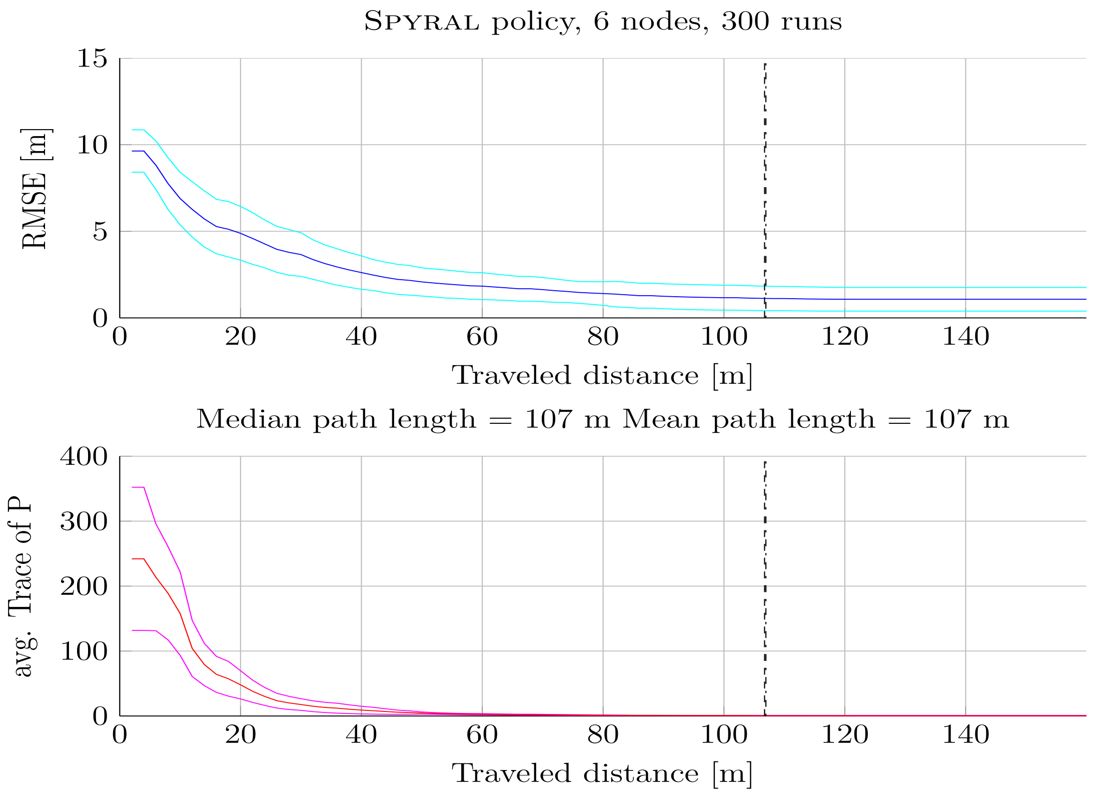
Table 5.1: Comparison between static and dynamic policies.

5.2.1 Static policies

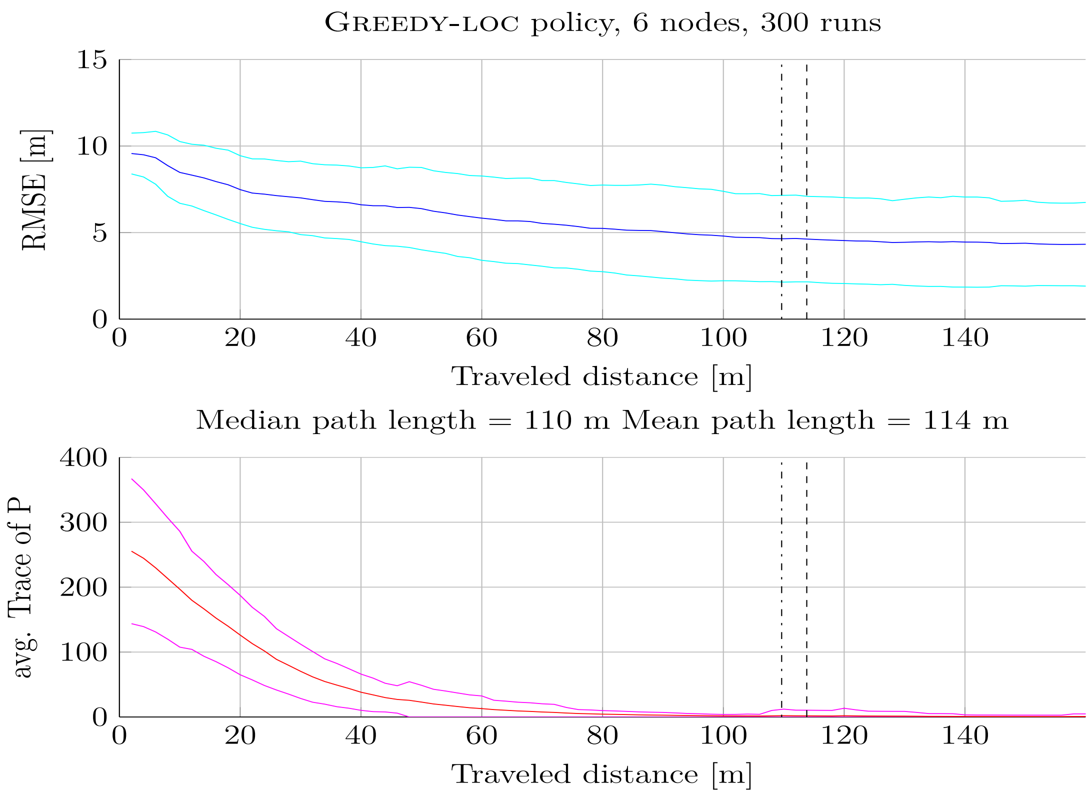


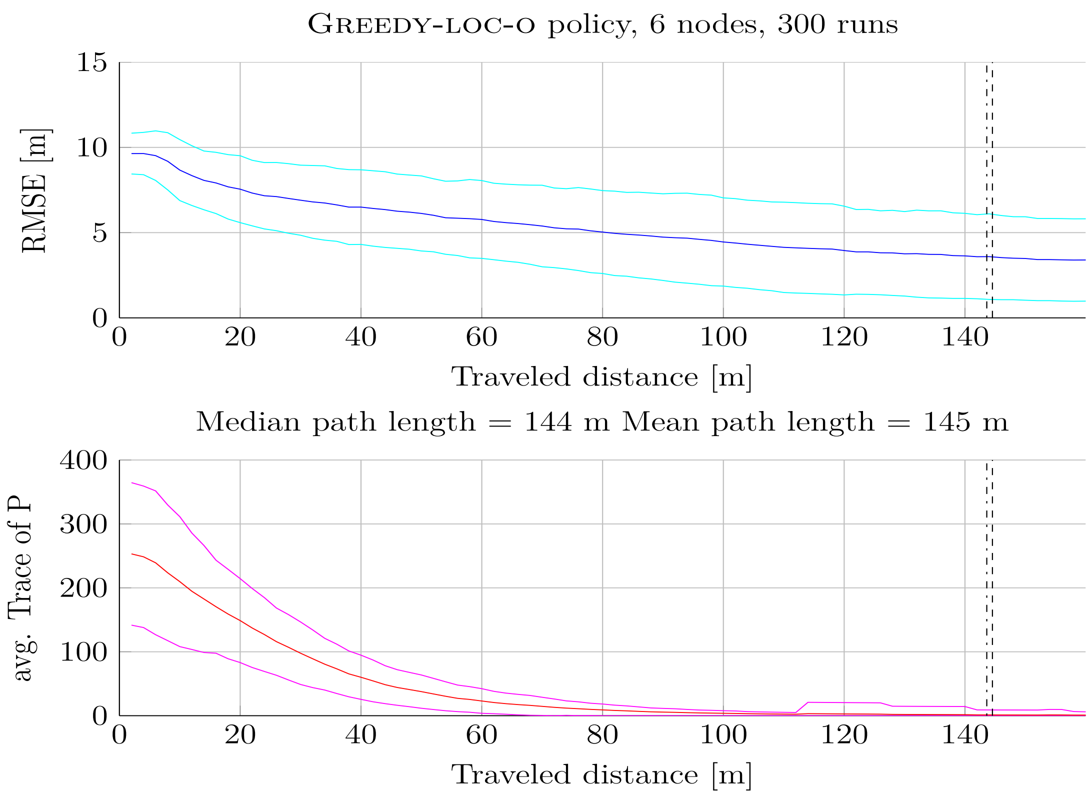


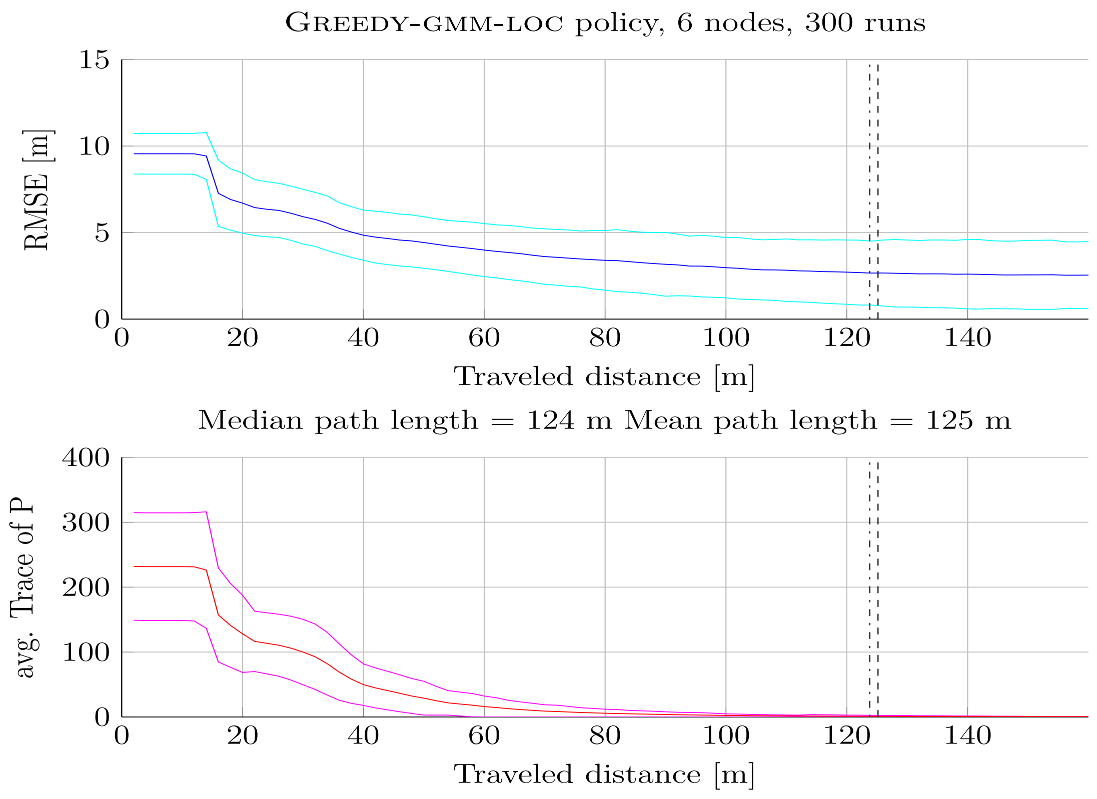

Squares policy, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 111 m Mean path length = 107 m
avg. Trace of P
Traveled distance [m]

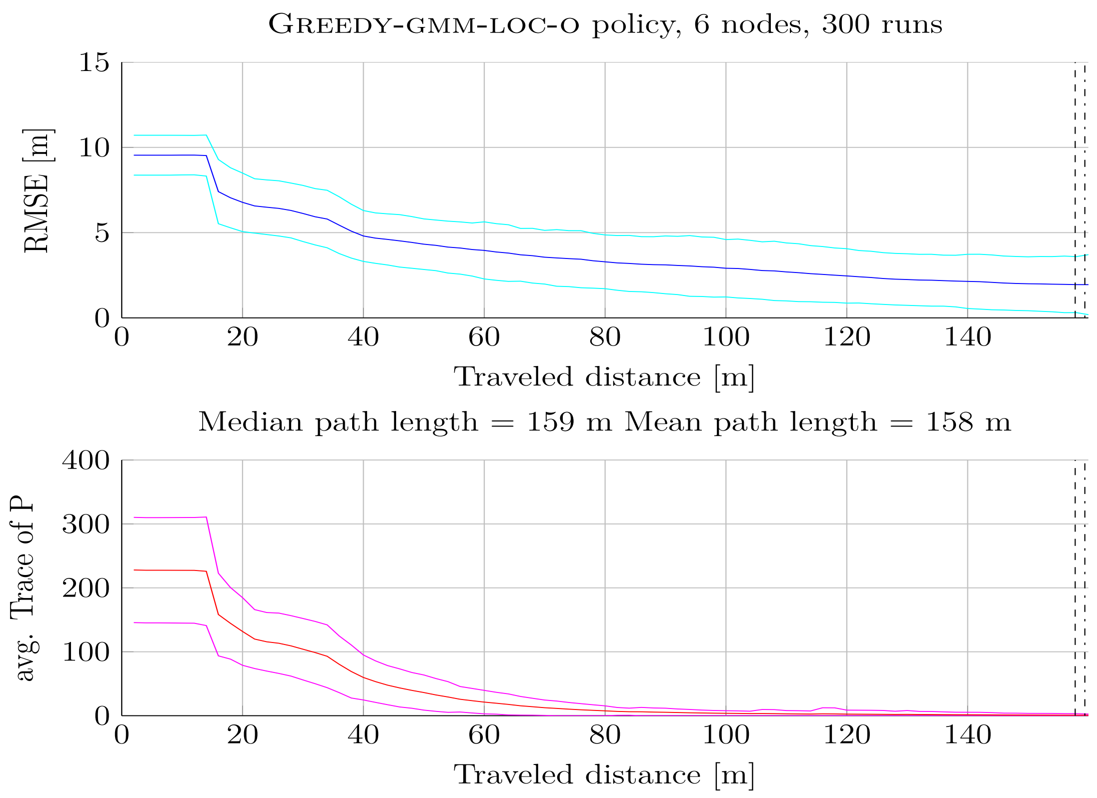

Spyral policy, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 107 m Mean path length = 107 m
avg. Trace of P
Traveled distance [m]

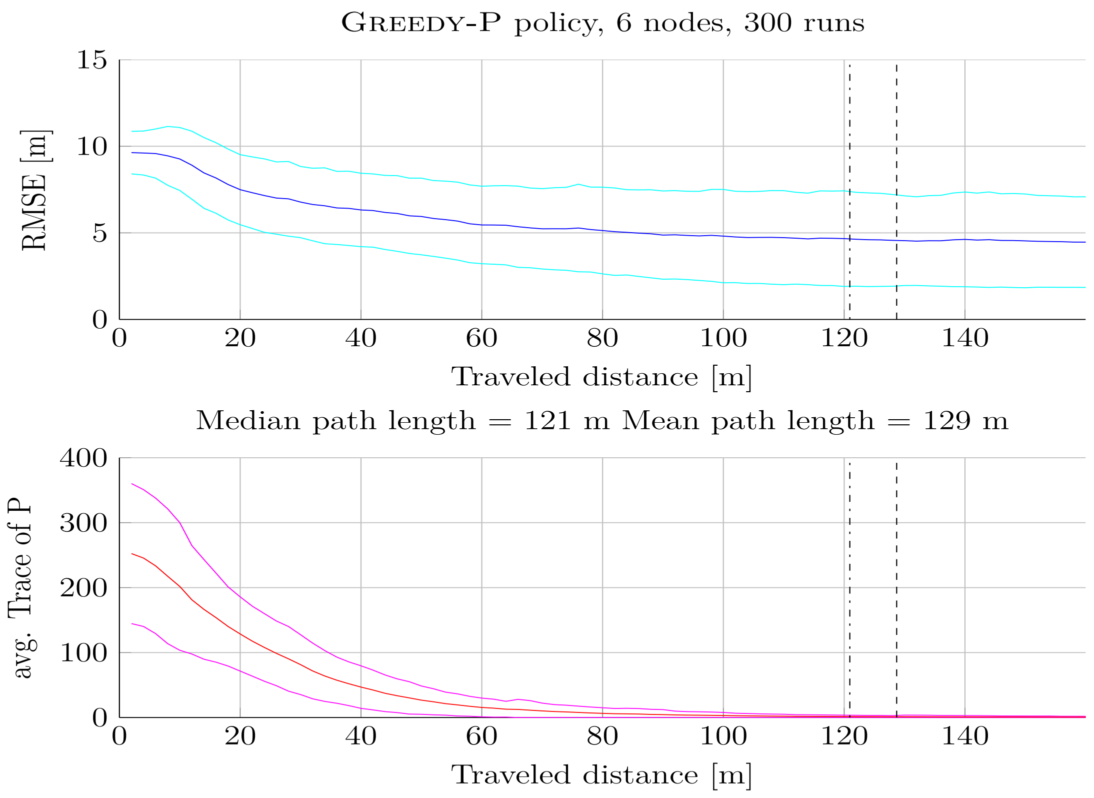
5.2.2 Greedy policies



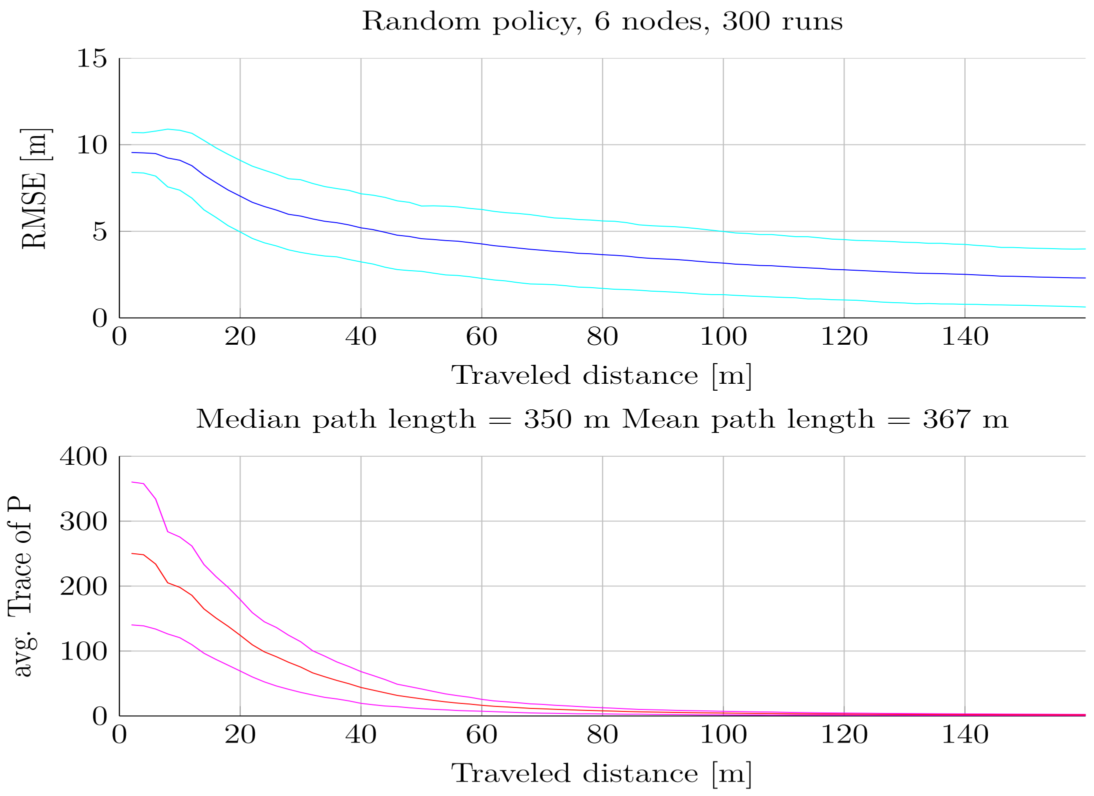



Greedy-gmm-loc policy, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 124 m Mean path length = 125 m
avg. Trace of P
Traveled distance [m]

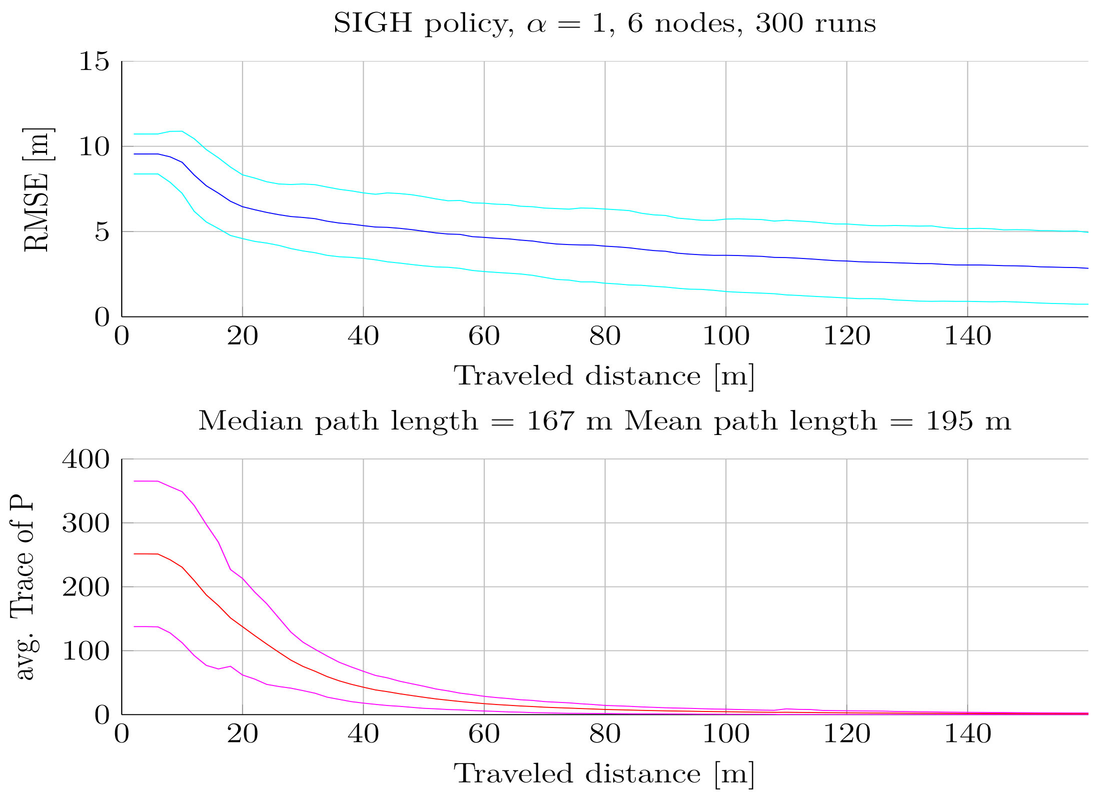

Greedy-gmm-loc-o policy, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 159 m Mean path length = 158 m
avg. Trace of P
Traveled distance [m]

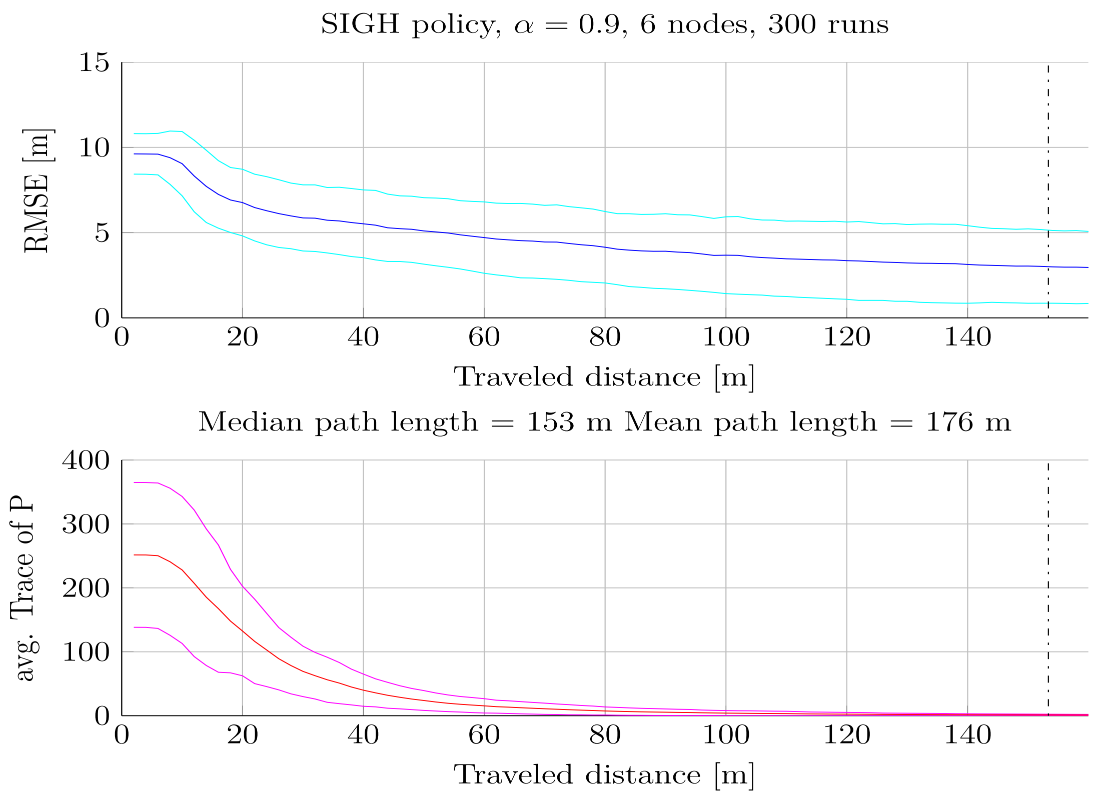


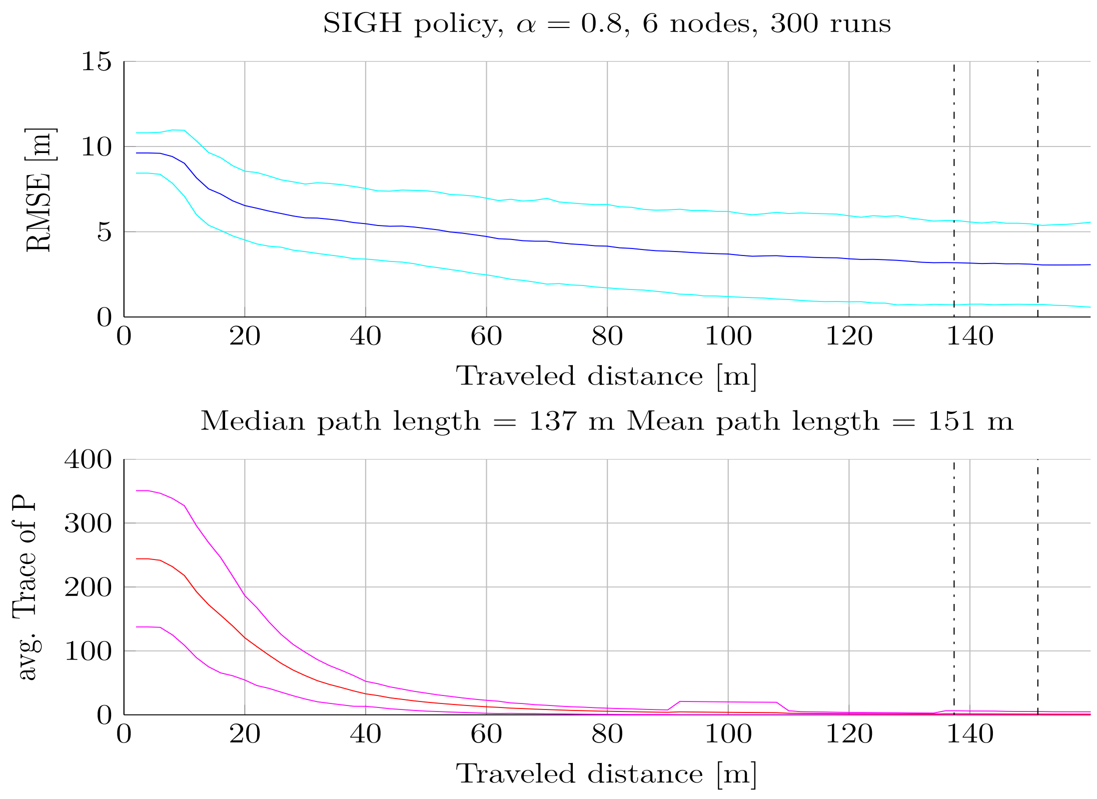
5.2.3 Random policy

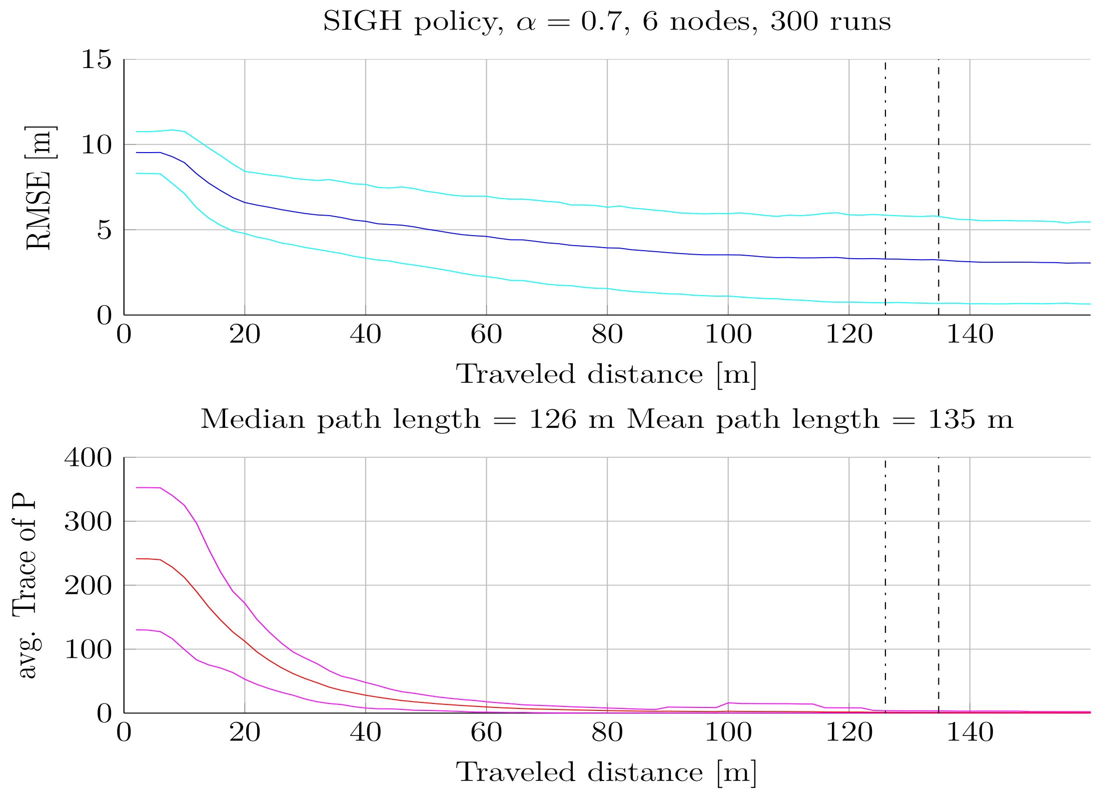


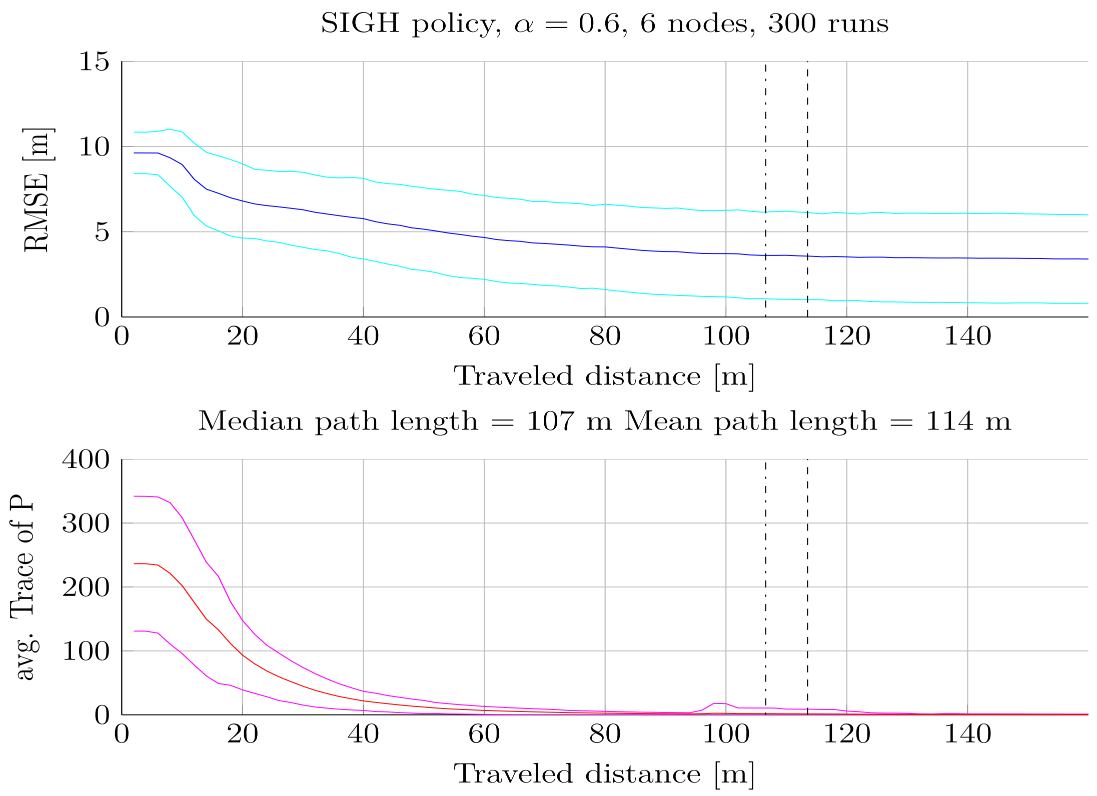
5.2.4 SIGH policy

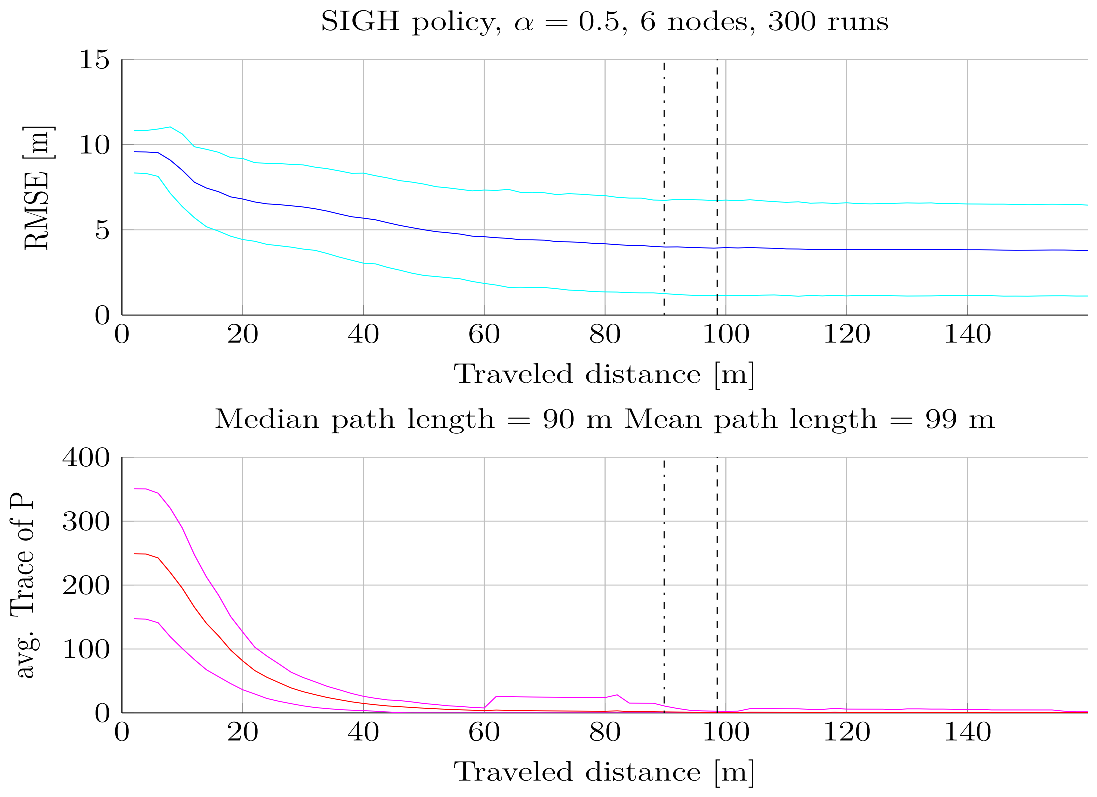





SIGH policy, $\alpha = 0.8$, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 137 m Mean path length = 151 m
avg. Trace of P
Traveled distance [m]


SIGH policy, $\alpha = 0.7$, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 126 m Mean path length = 135 m
avg. Trace of P
Traveled distance [m]


SIGH policy, $\alpha = 0.6$, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 107 m Mean path length = 114 m
avg. Trace of P
Traveled distance [m]


SIGH policy, $\alpha = 0.5$, 6 nodes, 300 runs
RMSE [m]
Traveled distance [m]
Median path length = 90 m Mean path length = 99 m
avg. Trace of P
Traveled distance [m]

Chapter 6

Conclusion

A complete solution for localizing a WSN with a mobile beacon was presented and implemented. It makes use of RSSI range measurements, an Unscented Kalman Filter, a mixture distribution for undelayed initialization and any one of the many proposed static and dynamic movement policies. Each and every one of those solutions tries to achieve low complexity in time and space, without sacrificing localization accuracy.

When choosing the best path to localize a random network, structure should be preferred to adaptability: the geometrical constraints enforced by static policies outperformed the erraticity of dynamic greedy algorithms. This resulted in shorter, more accurate, and more consistent paths.

Future research on the topic might highlight new movement strategies: extending the planning horizon to more than one time interval, or storing past measurements into a finite memory for delayed updates, could help introducing sound geometrical properties otherwise impossible to obtain for a dynamic memoryless policy.

Bibliography

[1] Bailey, T., and Durrant-Whyte, H. Simultaneous localization and mapping (slam): Part ii. *Robotics & Automation Magazine, IEEE 13*, 3 (2006), 108–117.

[2] Bardella, A., Bui, N., Zanella, A., and Zorzi, M. An experimental study on ieee 802.15. 4 multichannel transmission to improve rssi–based service performance. *Real-World Wireless Sensor Networks* (2010), 154–161.

[3] Bardella, A., Danieletto, M., Menegatti, E., Zanella, A., Pretto, A., and Zanuttigh, P. Autonomous robot exploration in smart environments exploiting wireless sensors and visual features. *Annals of Telecommunications* (2012), 1–15.

[4] Blumenthal, J., Grossmann, R., Golatowski, F., and Timmermann, D. Weighted centroid localization in zigbee-based sensor networks. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on* (2007), IEEE, pp. 1–6.

[5] Bourgault, F., Makarenko, A. A., Williams, S. B., Grocholsky, B., and Durrant-Whyte, H. F. Information based adaptive robotic exploration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on* (2002), vol. 1, IEEE, pp. 540–545.

[6] Caballero, F., Merino, L., and Ollero, A. A general gaussian-mixture approach for range-only mapping using multiple hypotheses. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (2010), IEEE, pp. 4404–4409.

[7] Chen, Y., Francisco, J.-A., Trappe, W., and Martin, R. P. A practical approach to landmark deployment for indoor localization. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on* (2006), vol. 1, IEEE, pp. 365–373.

[8] Fu, Q., Wei, C., Kezhong, L., Weibo, C., and Xiaoxi, W. Study on mobile beacon trajectory for node localization in wireless sensor networks. In *Information and Automation (ICIA), 2010 IEEE International Conference on* (2010), IEEE, pp. 1577–1581.

[9] Goldsmith, A. *Wireless communications*. Cambridge university press, 2005.

[10] Huang, G. P., Mourikis, A. I., and Roumeliotis, S. I. On the complexity and consistency of ukf-based slam. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 4401–4408.

[11] Huang, G. P., and Roumeliotis, S. I. An observability constrained ukf for improving slam consistency. *University of Minnesota, Minneapolis, MN, Tech. Rep* (2008).

[12] Huang, R., and Zaruba, G. Static path planning for mobile beacons to localize sensor networks. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on* (2007), IEEE, pp. 323–330.

[13] Julier, S., and Uhlmann, J. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE 92*, 3 (2004), 401–422.

[14] Julier, S. J. The scaled unscented transformation. In *American Control Conference, 2002. Proceedings of the 2002* (2002), vol. 6, IEEE, pp. 4555–4559.

[15] Julier, S. J., and Uhlmann, J. K. New extension of the kalman filter to nonlinear systems. In *AeroSense'97* (1997), International Society for Optics and Photonics, pp. 182–193.

[16] Julier, S. J., and Uhlmann, J. K. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. In

American Control Conference, 2002. Proceedings of the 2002 (2002), vol. 2, IEEE, pp. 887–892.

[17] Julier, S. J., Uhlmann, J. K., et al. A consistent, debiased method for converting between polar and cartesian coordinate systems. In *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, Florida. SPIE* (1997).

[18] Khan, U. A., Kar, S., and Moura, J. M. Distributed sensor localization in random environments using minimal number of anchor nodes. *Signal Processing, IEEE Transactions on 57*, 5 (2009), 2000–2016.

[19] Kim, K., Jung, B., Lee, W., and Du, D. Adaptive path planning for randomly deployed wireless sensor networks. *Journal of Information Science and Engineering 27*, 3 (2011), 1091–1106.

[20] Kim, K., and Lee, W. Mbal: A mobile beacon-assisted localization scheme for wireless sensor networks. In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on* (2007), IEEE, pp. 57–62.

[21] Koutsonikolas, D., Das, S., and Hu, Y. Path planning of mobile landmarks for localization in wireless sensor networks. *Computer Communications 30*, 13 (2007), 2577–2592.

[22] Krause, A., Singh, A., and Guestrin, C. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research 9* (2008), 235–284.

[23] Lefebvre, T., Bruyninckx, H., and De Schutter, J. Comment on "a new method for the nonlinear transformation of means and covariances in filters and estimators". *IEEE TRANSACTIONS ON AUTOMATIC CONTROL 47*, 8 (2002), 1407.

[24] Li, X., Mitton, N., Simplot-Ryl, I., and Simplot-Ryl, D. Dynamic beacon mobility scheduling for sensor localization. *Parallel and Distributed Systems, IEEE Transactions on 23*, 8 (2012), 1439–1452.

[25] Liu, J., Reich, J., and Zhao, F. Collaborative in-network processing for target tracking. *EURASIP Journal on Applied Signal Processing 2003* (2003), 378–391.

[26] Longford, N. T. Inference with the lognormal distribution. *Journal of Statistical Planning and Inference 139*, 7 (2009), 2329–2340.

[27] Menegatti, E., Danieletto, M., Mina, M., Pretto, A., Bardella, A., Zanconato, S., Zanuttigh, P., and Zanella, A. Autonomous discovery, localization and recognition of smart objects through wsn and image features. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE* (2010), IEEE, pp. 1653–1657.

[28] Menegatti, E., Danieletto, M., Mina, M., Pretto, A., Bardella, A., Zanella, A., and Zanuttigh, P. Discovery, localization and recognition of smart objects by a mobile robot. *Simulation, Modeling, and Programming for Autonomous Robots* (2010), 436–448.

[29] Menegatti, E., Zanella, A., Zilli, S., Zorzi, F., and Pagello, E. Range-only slam with a mobile robot and a wireless sensor networks. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 8–14.

[30] Merino, L., Caballero, F., and Ollero, A. Active sensing for range-only mapping using multiple hypothesis. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* (2010), IEEE, pp. 37–42.

[31] Nguyen, X., Jordan, M. I., and Sinopoli, B. A kernel-based learning approach to ad hoc sensor network localization. *ACM Transactions on Sensor Networks (TOSN) 1*, 1 (2005), 134–152.

[32] Picci, G. Filtraggio statistico (wiener, levinson, kalman) e applicazioni. *Libreria Progetto, Padova, Italy* (2007).

[33] Salo, J., Vuokko, L., El-Sallabi, H. M., and Vainikainen, P. An additive model as a physical basis for shadow fading. *Vehicular Technology, IEEE Transactions on 56*, 1 (2007), 13–26.

[34] Shachter, R. D. Bayes-ball: Rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the fourteenth conference on Uncertainty in Artificial Intelligence* (1998), Morgan Kaufmann Publishers Inc., pp. 480–487.

[35] Sichitiu, M., and Ramadurai, V. Localization of wireless sensor networks with a mobile beacon. In *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on* (2004), IEEE, pp. 174–183.

[36] Sim, R., and Roy, N. Global a-optimal robot exploration in slam. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* (2005), IEEE, pp. 661–666.

[37] Sola, J., Monin, A., Devy, M., and Lemaire, T. Undelayed initialization in bearing only slam. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on* (2005), IEEE, pp. 2499–2504.

[38] Srinath, T. Localization in resource constrained sensor networks using a mobile beacon with in-ranging. In *Wireless and Optical Communications Networks, 2006 IFIP International Conference on* (2006), IEEE, pp. 5–pp.

[39] Stachniss, C., Grisetti, G., and Burgard, W. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of robotics: science and systems (RSS)* (2005), pp. 65–72.

[40] Sun, G.-L., and Guo, W. Comparison of distributed localization algorithms for sensor network with a mobile beacon. In *Networking, Sensing and Control, 2004 IEEE International Conference on* (2004), vol. 1, IEEE, pp. 536–540.

[41] Teng, G., Zheng, K., and Yu, G. A mobile-beacon-assisted sensor network localization based on rss and connectivity observations. *International Journal of Distributed Sensor Networks 2011* (2011).

[42] Torgerson, W. Theory and methods of scaling.

[43] Wan, E., and Van Der Merwe, R. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communica-*

tions, and Control Symposium 2000. AS-SPCC. The IEEE 2000 (2000), IEEE, pp. 153–158.

[44] Wang, H., Yao, K., Pottie, G., and Estrin, D. Entropy-based sensor selection heuristic for target localization. In *Proceedings of the 3rd international symposium on Information processing in sensor networks* (2004), ACM, pp. 36–45.

[45] Wikipedia. Log-normal distribution — wikipedia, the free encyclopedia, 2013. [Online; accessed 17-March-2013].

[46] Wu, Y., Hu, D., Wu, M., and Hu, X. Unscented kalman filtering for additive noise case: augmented vs. non-augmented. In *American Control Conference, 2005. Proceedings of the 2005* (2005), IEEE, pp. 4051–4055.

[47] Xiao, B., Chen, H., and Zhou, S. Distributed localization using a moving beacon in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on 19*, 5 (2008), 587–600.

[48] Yap, P. Grid-based path-finding. In *Advances in Artificial Intelligence*. Springer, 2002, pp. 44–55.

[49] Yong, W., Xiaobu, X., and Xiaoling, T. Localization in wireless sensor networks via support vector regression. In *Genetic and Evolutionary Computing, 2009. WGEC'09. 3rd International Conference on* (2009), IEEE, pp. 549–552.

[50] Zanella, A., Menegatti, E., and Lazzaretto, L. Self-localization of wireless sensor nodes by means of autonomous mobile robots. In *Wireless Communications 2007 CNIT Thyrrenian Symposium* (2007), Springer, pp. 309–320.