Analysis of two approaches to location estimation based on wireless signal strength propagation and Gaussian Processes

○ ミヤグスク レナート (東京大学) 正 山下 淳（東京大学）
正 淺間 一（東京大学）

Renato Miyagusuku, The University of Tokyo, miyagusuku@robot.t.u-tokyo.ac.jp
Atsushi Yamashita, The University of Tokyo
Hajime Asama, The University of Tokyo

Robot localization is the problem of determining a robot's pose based on sensory information. This problem is consider one of the fundamental issues for autonomous robotics, hence its importance. Localization systems using wireless signal strength measurements have gained popularity in recent years, probably due to the proliferation of wireless Local Area Networks using Wi-Fi. Among these systems Gaussian Processes excel due to its flexibility and ability to model a wide variety of mappings. This paper presents an analysis and comparison of two different approaches for the localization problem based on wireless signal strength measurements. Our main motivation is the use of these learned mappings for robust localization.

Key Words: mapping, Localization, Gaussian Processes

1 Introduction

Robot localization is considered a key problem in making a robot trully autonomous, as robots need to estimate their pose within a fixed coordinate system in order to determine what action to do next.This pose has to be estimated based on sensory information, popular sensors used for location include GPS, laser range finders, RGB-D cameras, sonars, among others. The adition of new sensors usually implies an improvement on precision and robustness - thanks to redundancy - of the approach, if well implemented. Thanks to the proliferation of wireless Local Area Networks, localization systems using wireless signal strength measurements have become a compelling choice, as the hardware set-up (the wireless network) is already in place in most environments. Among the methods used for signal strength-based location, Gaussian Processes excel due to its flexibility and ability to model a wide variety of mappings. Gaussian Processes are flexible non-parametric distributions used for learning input-output mappings. They are fully defined by a mean and a covariance function. Previous research has shown promising results in signal strength-based location estimation using Gaussian Processes [1], even without explicit knowledge of the positions were the data was taken [2]. Gaussian Processes can be classified as a fingerprinting technique. Fingerprinting algorithms collect features (fingerprints) of in a map and then estimate the location of an object by matching new measurements to previously acquired data. One of the first works to successfully employ fingerprinting for wireless signal strength-based localization was RADAR [3]. Other classifications for wireless signals-based location techniques are triangulation, and proximity - being Place-Lab [4] possibly the most well know representative. A more comprehensive review on previously developed systems can be found at [5].

2 Gaussian Processes

As explained by Rasmussen [6] Gaussian Processes are a generalization of normal distributions to functions, describing functions of finite-dimensional random variables. In a nutshell, given some training points, a Gaussian Process generalizes this points into a continuous function where each point is considered to have normal distribution, hence a mean and a variance. The essence of the method resides is assuming a correlation between values at different points, this correlation is characterized by a covariance function or a kernel.

Formally, given some training data $(\mathbf{X}, \mathbf{y})$ where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the matrix of n input samples $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^n$ the vector of corresponding outputs $y_i \in \mathbb{R}$. Two assumptions are made. First, each data pair $(\mathbf{x}_i, \mathbf{y}_i)$ is assumed to be drawn from a noisy process:

$$\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon, \qquad (1)$$

where ϵ is the noise generated from a Gaussian distribution with known variance σ_n^2.

Second, any two output values, $\mathbf{y}_p$ and $\mathbf{y}_q$, are assumed to be correlated by a covariance function based on their input values $\mathbf{x}_p$ and $\mathbf{x}_q$:

$$cov(\mathbf{y}_p, \mathbf{y}_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \qquad (2)$$

where $k(\mathbf{x}_p, \mathbf{x}_q)$ is a kernel, σ_n^2 the variance of ϵ and δ_{pq} is one if $p = q$ and zero otherwise.

Given these assumptions, for any finite number of data points, the Gaussian Process can be considered to have a multivariate Gaussian distribution:

$$\mathbf{y} \sim \mathcal{N}(m(\mathbf{x}), cov(\mathbf{x}_p, \mathbf{x}_q)) \qquad (3)$$

and therefore be fully defined by a mean function $m(\mathbf{x})$ and a covariance function $cov(\mathbf{x}_p, \mathbf{x}_q)$. However, it is common for $m(\mathbf{x})$ to be set to zero, in which case it is only necessary to define the kernel of the covariance function in order to fully describe the Gaussian Process. It is important to notice that setting the mean to zero does not imply that at an arbitrary point $\mathbf{x}_*$, the function $f(\mathbf{x}_*)$ will be zero, it simply states that given no information, the expectation of the point will be zero.

2.1 Kernels

Kernels are functions that depict the relationship between any to inputs $\mathbf{x}_p$ and $\mathbf{x}_q$. Any function can not be a kernel, in general the function must always yield positive semi-definite covariance matrices.

The perhaps most popular kernel function is the squared exponential kernel, also know as Gaussian or radial basis function kernel. Which is defined as:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{(\mathbf{x}_p - \mathbf{x}_q)^2}{2l^2}\right), \tag{4}$$

with hyper-parameters: variance (σ_f^2) and lengthscale = l.

2.2 Prediction

For the prediction of $\mathbf{f}_* = f(\mathbf{x}_*)$ of an unknown data points $\mathbf{x}_*$, eq. (3) can be written as:

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}\mathbf{K} + \sigma_n^2\mathbf{I}_n & \mathbf{K}_* \\ \mathbf{K}_*^T & k(\mathbf{x}_*, \mathbf{x}_*)\end{bmatrix}\right), \tag{5}$$

where $\mathbf{K}$ is $k(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_*$ is $k(\mathbf{X}, \mathbf{x}_*)$ and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Now, conditioning $\mathbf{f}_*$ to all other variables ($\mathbf{x}_*$, $\mathbf{X}$, $\mathbf{y}$), we obtain a probability distribution of $\mathbf{f}_*$:

$$\mathbf{f}_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y} \ \sim \ \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)), \tag{6}$$

where,

$$\bar{\mathbf{f}}_* = \mathbf{K}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I}_n)^{-1}\mathbf{y}, \tag{7}$$

$$cov(\mathbf{f}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I}_n)^{-1}\mathbf{K}_*. \tag{8}$$

In case of multiple outputs, that is training data $(\mathbf{X}, \mathbf{Y})$ where $\mathbf{Y} \in \mathbb{R}^{n \times m}$ is the matrix of m output samples $\mathbf{y}$, a different Gaussian Process can be generated for each vector $\mathbf{y}$, but more commonly than not all vectors are generated using the same Gaussian Process, that is the same covariance function. Therefore eq. (7) simply becomes:

$$\bar{\mathbf{F}}_* = \mathbf{K}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I}_n)^{-1}\mathbf{Y}, \tag{9}$$

and eq. (8) remains the same.

It is thanks to this ease of conditioning variables that Gaussian Processes are so powerful.

2.3 Optimization

What is left now it to optimize the covariance function for our training data. Considering θ a set of hyper-parameters that define the covariance function, and the training points $(\mathbf{X}, \mathbf{y})$. The maximum a posteriori estimation of θ occurs when we maximize $\theta|\mathbf{X}, \mathbf{y}$ From Bayes, we have that:

$$\theta|\mathbf{X}, \mathbf{y} \ \propto \mathbf{y}|\mathbf{X}, \theta \times \theta|\mathbf{X} \tag{10}$$

However, if the a priori distribution $p(\theta|\mathbf{X})$ holds little to no information, the following also holds:

$$\theta|\mathbf{X}, \mathbf{y} \ \propto \mathbf{y}|\mathbf{X}, \theta \tag{11}$$

Therefore, maximizing $\theta|\mathbf{X}, \mathbf{y}$ is also equivalent to minimizing the negative log likelihood (nll) of $\mathbf{y}|\mathbf{X}, \theta$

$$nll = 0.5 \log|\mathbf{K} + \sigma_n\mathbf{I}_n| + 0.5\mathbf{y}^T(\mathbf{K} + \sigma_n\mathbf{I}_n)^{-1}\mathbf{y} + 0.5n\log(2\pi) \tag{12}$$

In case of multiple outputs under the same covariance function, the nll becomes the summation of the nll for each output vector $\mathbf{y}$

3 Modeling wireless signal strength

Using Gaussian Processes to solve the location estimation problem using wireless strength measurements we have a choice to make, either using signal strength measurements as inputs and obtaining estimated positions as outputs; or using position information as inputs and obtaining estimated signal strength measurements as outputs - which can be used to estimate the likelihood of a signal strength sample being originated from any given position, and solving the problem with a particle filter or any other similar localization algorithm. The first approach is straight forward, the goal of the localization algorithm is to find suitable position estimations based on signal strength samples, and that is exactly what the Gaussian Process will do. However, as we will show in the following sections, the second option, although more complex, is more robust.

Formally, given some training data $(\mathbf{Z}, \mathbf{P})$ where $\mathbf{Z} \in \mathbb{R}^{n \times n_{AP}}$ is the matrix of n signal strength samples $\mathbf{z}_i \in \mathbb{R}^{n_{AP}}$ corresponding to n_{AP} different access points present in the environment, and $\mathbf{P} \in \mathbb{R}^{n_AP \times 2}$ is the matrix composed by the corresponding vectors $\mathbf{p}_i \in \mathbb{R}^2$ of the x, y positions where the measurements were taken.

3.1 First approach

The first approach considers generating a Gaussian Process with inputs $\mathbf{Z}$ and outputs $\mathbf{P}$, and use it to predict unknown positions $\mathbf{P}_*$ given new signal strength samples $\mathbf{z}_*$, from eq. (7) we have that the expectation for the unknown positions would be:

$$\bar{\mathbf{P}}_* = \mathbf{K}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I}_n)^{-1}\mathbf{P}. \tag{13}$$

Given enough training points, this approach efficiently predicts the location of new signal strength samples. Furthermore, as it is a straight forward approach to the location estimation problem, it is faster than the second one. However the main issue with this approach is its robustness. As it will be shown in the next section, when signal samples are scaled or the signal strength sample for a particular access point is missing, the performance of the system can be severely affected.

3.2 Second approach

The second approach considers generating a Gaussian Processes with inputs $\mathbf{P}$ and outputs $\mathbf{Z}$, and use it to generate signal strength probability distributions $\mathbf{z}|\mathbf{p}$. First, we can compute $\mathbf{z}_*$ for any $\mathbf{p}_* \in \mathbf{p}$, based on eqs. (7,8):

$$\bar{\mathbf{z}}_* = \mathbf{K}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I}_n)^{-1}\mathbf{Z}, \tag{14}$$

$$cov(\mathbf{z}_*) = k(\mathbf{p}_*, \mathbf{p}_*) - \mathbf{K}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I}_n)^{-1}\mathbf{K}_*, \tag{15}$$

so from the formula for multivariate Gaussian distribution we have:

$$\mathbf{z}|\mathbf{p} = (2\pi * cov(\mathbf{z}_*))^{-\frac{n_{AP}}{2}} \exp\left(-(\mathbf{z} - \mathbf{z}_*)^T(cov(\mathbf{z}_*))^{-1}(\mathbf{z}_*)\right), \tag{16}$$

we will also define log $\mathbf{z}|\mathbf{p}$ as it will be also used for visualization in the next section:

$$\log\ \mathbf{z}|\mathbf{p} = -0.5n_{AP}\log(2\pi * cov(\mathbf{z}_*)) - (\mathbf{z} - \mathbf{z}_*)^T(cov(\mathbf{z}_*))^{-1}(\mathbf{z} - \mathbf{z}_*). \tag{17}$$

Given enough training points, this approach also efficiently predicts the location of new signal strength samples. Furthermore, scaling of signal samples almost do not affect its performance, nor do missing.

4 Simulations

For the implementation of both approaches we used the open source library GPy [7], using an rbf kernel - same as kernel described in subsection 2.1, with hyper-parameters rbf.variance and rbf.lengthscale and gaussian_noise, corresponding respectively to σ_f,l, and σ_n; and the optimization was performed using an LBFGS gradient descend algorithm. The dataset used for testing was an open source dataset[1] with signal measurements obtained from a tour around University of Washington Paul Allen building [2], consisting of measurements for 30 access points for 215 training points and 179 testing points. A value of 92 dBm was added to the true measured values and then scaled by a factor of 15. Failing to get any data in a mesurement, or acquiring values lower than 90 dBm, defaults to zero. This is in consideration of typical wireless network controller sensitivity of up to -90 dBm. Positions were measured in meters. For all simulations the complete set of training points was used for optimization, and the set of testing points was used for testing. To asses the robustness of the method we considered two possible cases. First, the scaling of signal strength measurements by an arbitrary factor; and second, the setting of 0. to all measurements originated from some specific access point. The first case can occur when the hardware used for taking the samples is changed; or if a constant object such as an additional sensor is mounted on the robot, obstructing incoming signals. In order to test the robustness of the system in this particular case, each approach was tested using testing points scaled by 0.8 and inspecting its new performance. The second case can temporarily occur by occlusions in the line of sight between transmitter and receptor or by random anomalies in the signal propagation, it can also occur permanently in case of access point malfunction. This case is tested by artificially setting the values of access points 15 and 20 to zero.

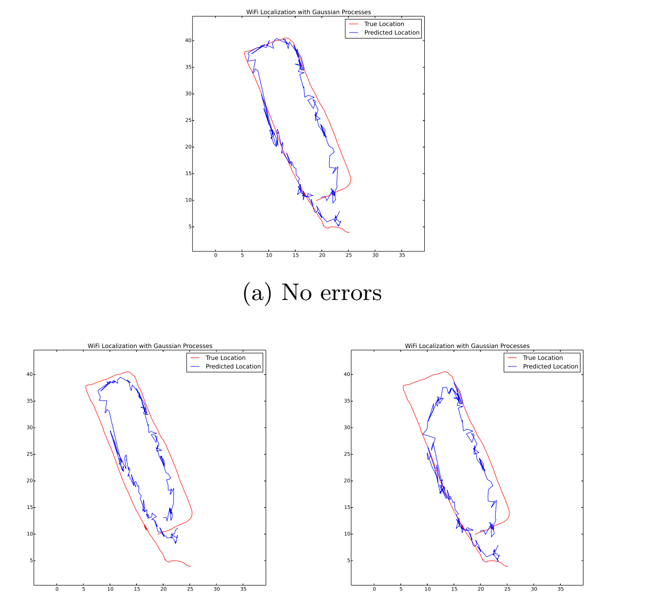
4.1 First approach

Using the training points from the first dataset a Gaussian Process following the description of the first approach was optimized. The resulting hyper-parameters were:

rbf.variance = 81.54
rbf.lengthscale = 4.79
gaussian_noise = 0.244

This model was used for all simulations, as the errors are supposed to be induced after the system has already been trained.

Figure 1a shows the output of the system when fully operational. Blue lines represents $\mathbf{p}_*$ and red ones the ground truth. As we can see from Fig. 1b, for the first error case, the performance of the estimation drops at all testing points, showing that this approach is quite susceptible to scaling. Figure 1c shows the output for the second error case, where the system's performance evidently drops for test points near the upper left corner. However, for all other regions, the systems seems only slightly affected.



(a) No errors

(b) Signal strength measurements scaled by 0.8

(c) Access points 15 and 20 values set to zero

Fig.1: Expected positions obtained using the first approach. Given some testing points, blue lines represents estimated positions while red ones the ground truth.

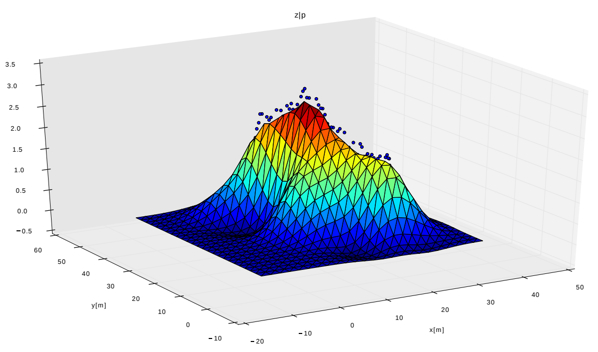


Fig.2: Expected values. Blue dots represent signal strength samples, while the surface shows the learned mapping expected values

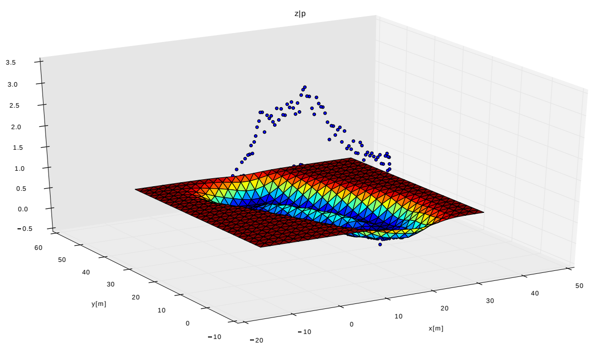
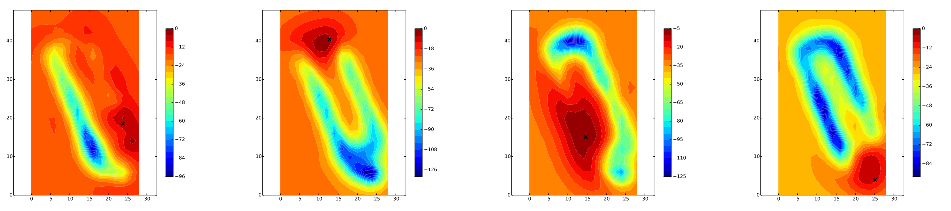
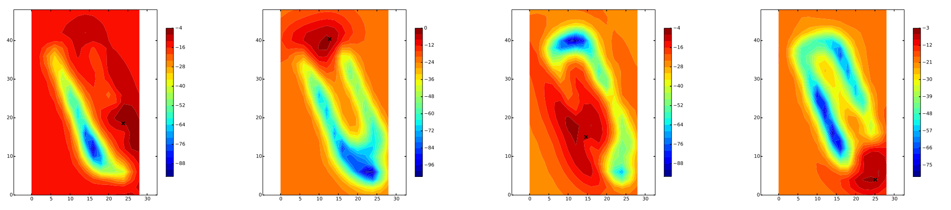


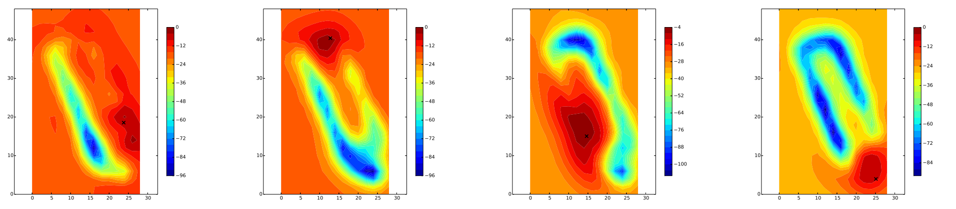
Fig.3: Covariance values. Blue dots represent signal strength samples, while the surface shows the learned mapping covariance values



(a) Log likelihood of $\mathbf{z}_*|\mathbf{p}$ for different test points



(b) Log likelihood of $\mathbf{z}_*|\mathbf{p}$ for different test points with measurements scaled by 0.8



(c) Log likelihood of $\mathbf{z}_*|\mathbf{p}$ for different test points with measurement of access points 15 and 20 set to 0.

Fig.4: Maps show the log likelihood of a signal sample $\mathbf{z}_*$ for every position $\mathbf{p}_i$ in the (x,y) coordinates. The black **X** represents the testing point position and the contour plot the log likelihood of the signal sampled at that point. Red values indicate high probabiliy while blue ones indicate low

4.2 Second approach

Using the same training points from the first dataset a Gaussian Process following the description of the second approach was optimized. The resulting hyper-parameters were:

rbf.variance = 0.526
rbf.lengthscale = 4.46
gaussian_noise = 0.125

Figures 2 and 3 show mappings generated by the optimized Gaussian Process for one access points - a pair of such mappings was generated for each acces point. Using all these, the log likelihood for signal strength measurements at any point of the map were computed by eq. (17). Figure 4a shows the likelihoods at 4 different arbitrarily selected test points (25,73,125,175), when no errors in the measurement are considered. For the first error case Fig. 4b shows that although the log likelihood diminishes, it does so almost equally for all points **p**, therefore, the systems performance in general is not that affected. Same as with the error case 2 shown at Fig. 4c.

5 Conclusions

Simulations have shown the feasibility of using Gaussian Processes to solve the signal strength-based location estimation problem. Between the two proposed approaches, the second one has been proved to be more robust. Nonetheless, it is important to mention that the first approach's robustness against the second error scenario is comparatively similar, and that it is faster as its outputs solve the location estimation problem without requiring any additional algorithm, such as a particle filter - and the additional computations this involves.

6 Acknowledgments

This work was in part supported by Tough Robotics Challenge, ImPACT Program (Impulsing Paradigm Change through Disruptive Technologies Program).

References

[1] Brian Ferris, Dirk Haehnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *In proc. of robotics science and systems*. Citeseer, 2006.

[2] Brian Ferris, Dieter Fox, and Neil D Lawrence. Wifi-slam using gaussian process latent variable models. In *IJCAI*, volume 7, pages 2480–2485, 2007.

[3] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000.

[4] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. Place lab: Device positioning using radio beacons in the wild. In *Pervasive computing*, pages 116–133. Springer, 2005.

[5] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.

[6] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for machine learning. *the MIT Press*, 2(3):4, 2006.

[7] The GPy authors. GPy: A gaussian process framework in python, 2012–2014.

[1]https://github.com/sods/ods