Bernstein Center for
Computational Neuroscience
Berlin

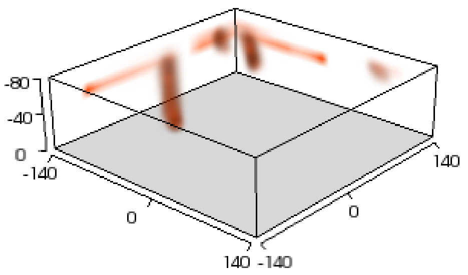
Master Thesis

Computational Neuroscience

# Gaussian Processes for Plume Distribution Estimation with UAVs

Jan Gosmann

December 2, 2013



Supervisors:
Prof. Dr. Manfred Opper
Dr. Andreas Ruttor

## Eidesstattliche Versicherung

Hiemit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 2. Dezember 2013

Jan Gosmann

## Abstract

Recent scientific work explored the possibility to use mobile robots for environmental monitoring. This includes for example the estimation of ozone concentrations or locating the source of a pollutant plume. So far the modeling of the complete spatial distribution of a plume (which has different spatial characteristics compared to the ozone concentrations) has not been done. In this work existing methods of Bayesian optimization, namely global optimization (GO) and the distance based upper confidence bound (DUCB), were evaluated on this task. Also, a new method – plume distance based upper confidence bound (PDUCB) – and an extension to multiple robots is proposed. All methods were tested in simulations using the QRSim quadrotor simulator. The existing methods were not able to solve the task satisfyingly, whereas the PDUCB method was able to approximate plume distributions with noisy measurements reasonably well.

## Zusammenfassung

Neuere wissenschaftliche Arbeiten haben die Möglichkeit untersucht mobile Roboter zur Umweltüberwachung einzusetzen. Dies beinhaltet die Vermessung von Ozonkonzentrationen und die Lokalisierung der Quelle einer Gasfahne. Bisher wurde jedoch nicht versucht die komplette räumliche Verteilung einer Gasfahne (welche andere räumliche Charisitika im Vergleich zu Ozonkonzentrationen hat) zu bestimmen. In dieser Arbeit wurde die Anwendbarkeit bestehender Methoden Bayes'scher Optimierung, globale Optimierung (GO) und die distanzbasierte obere Konfidenzgrenze (DUCB), für diese Aufgabe beurteilt. Weiterhin wird eine neue Methode – die distanzbasierte obere Konfidenzgrenze für Gasfahnen (PDUCB) – sowie eine Erweiterung für mehrere Roboter vorgeschlagen. Alle Methoden wurden in Simulationen mit dem QRSim Quadrotorsimulator getestet. Die bestehenden Methoden waren nicht in der Lage die Aufgabe zufriedenstellend zu lösen, wohingegen die PDUCB-Methode in der Lage war die Verteilungen der Gasausbreitung angemessen gut mit verrauschten Messungen anzunähern.

# Contents

# Symbols and Notation

Variables are typeset in italics, whereas constants are upright. Vectors and matrices use a bold font. Additionally, matrices use uppercase letters.

| | |
|---|---|
| $[\,]$ | matrix in block notation |
| $\sim$ | distributed according to |
| $\bar{x}$ | mean of the random variable $x$ |
| $x\|y$ | conditional random variable $x$ given $y$ |
| $\|\boldsymbol{x}\|$ | Euclidean ($L^2$) norm of $\boldsymbol{x}$ |
| $\mathbf{0}$ | null vector $(0,\dots,0)^\top$ |
| $\\|\boldsymbol{A}\\|$ | norm of $\boldsymbol{A}$ |
| $(\boldsymbol{A})_{ij}$ | element of $\boldsymbol{A}$ at row $i$ and column $j$ |
| $\boldsymbol{A}^\top$ | the transpose of $\boldsymbol{A}$ |
| $\mathcal{BV}$ | set of basis vectors |
| $\mathrm{cov}(\boldsymbol{x})$ | covariance (matrix) of elements of $\boldsymbol{x}$ |
| $-\boldsymbol{C_t}$ | sparsely approximated covariance matrix (with $t$ samples) |
| $c(\boldsymbol{x})$ | true concentration distribution |
| $\mathcal{D}$ | set of combined training inputs and targets $(\boldsymbol{x}_i, y_i)$ |
| $\det \boldsymbol{A}$ | the determinant of $\boldsymbol{A}$ |
| $\mathrm{diag}\, \boldsymbol{A}$ | vector with the diagonal elements of the matrix $\boldsymbol{A}$ |
| $\mathrm{diag}\, \boldsymbol{x}$ | diagonal matrix with the elements of the vector $\boldsymbol{x}$ |
| $d(\boldsymbol{x}, \boldsymbol{x}')$ | Euclidean distance (unitless) |
| $E$ | error measure |
| $\hat{E}$ | estimate of an error measure |
| $F$ | normalized error |
| $\tilde{\boldsymbol{K}}$ | covariance matrix including the noise variance $\sigma_\mathrm{n}^2$ on the diagonal |
| $K(X, X')$ | matrix of pairwise covariances of $\boldsymbol{x} \in X$ and $\boldsymbol{x}' \in X'$ |
| $k(\boldsymbol{x}, \boldsymbol{x}')$ | kernel or covariance function |
| $K_\nu(z)$ | modified Bessel function |
| $\boldsymbol{L}$ | Cholesky factor (lower, triangular matrix) |
| $\ln$ | natural logarithm |
| $m(\boldsymbol{x})$ | mean function of a Gaussian process |
| $\mathrm{mslim}$ | mean square limit |
| $N(x; \mu, \sigma^2)$ | Gaussian probability density at $x$ with mean $\mu$ and variance $\sigma^2$ |
| $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ | (multivariate) normal distribution with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$ |

| | |
|---|---|
| $p(x)$ | probability or probability density of $x$ |
| $R$ | QRSim reward |
| $s(\boldsymbol{y})$ | Scaling factor in an acquisition function |
| $u(\boldsymbol{x})$ | utility function |
| $V$ | the simulated volume |
| $X$ | set of training inputs |
| $\boldsymbol{x}$ | some location or input data |
| $X_*$ | set of test inputs |
| $x_i$ | $i$-th component of the vector $\boldsymbol{x}$ |
| $\boldsymbol{y}$ | vector of training targets |
| $\Gamma(\nu)$ | Gamma function |
| $\Phi(x; \mu, \sigma^2)$ | Gaussian cummulative distribution function at $x$ with mean $\mu$ and variance $\sigma^2$ |
| $\gamma$ | distance penalty weighting |
| $\kappa$ | variance weighting |
| $\mu(\boldsymbol{x})$ | mean predicted by a Gaussian process (unitless) |
| $\rho$ | weighting of distance to other UAVs |
| $\sigma^2(\boldsymbol{x})$ | variance predicted by a Gaussian process (unitless) |
| $\sigma_{\mathrm{k}}^2$ | kernel process variance |
| $\sigma_{\mathrm{n}}^2$ | noise variance of a Gaussian process |
| $\sigma_{\mathrm{sn}}^2$ | sensor noise variance |
| $\sigma_{\mathrm{t}}^2$ | novelty of input $\boldsymbol{x}_t$ |

# Acronyms

| | |
|---|---|
| BO | Bayesian optimization |
| D-NF-MS-SV | Gaussian dispersion, noise free, multiple source, single vehicle scenario |
| D-NF-SS-SV | Gaussian dispersion, noise free, single source, single vehicle scenario |
| D-SN-MS-MV | Gaussian dispersion, sensor noise, multiple source, multiple vehicle scenario |
| D-SN-MS-SV | Gaussian dispersion, sensor noise, multiple source, single vehicle scenario |
| D-SN-SS-SV | Gaussian dispersion, sensor noise, single source, single vehicle scenario |
| DUCB | distance-based upper confidence bound |
| EU | European Union |
| G-NF-SS-SV | Gaussian, noise free, single source, single vehicle scenario |
| GPS | Global Positioning System |
| IEM | intelligent environmental monitoring |
| IMU | inertial measurement unit |
| KDE | kernel density estimation |
| MH | Metropolis-Hastings |
| MS | mean square |
| NED | north, east, down |
| norm. | normalized |
| PDUCB | plume distance-based upper confidence bound |
| RBF | radial basis function |
| RMISE | root mean integrated square error |
| SD | standard deviation |
| SE | squared exponential |
| UAV | unmanned aerial vehicle |
| WRMISE | weighted root mean integrated square error |

# 1. Introduction

Environmental monitoring is used to ensure water and air pollution levels are in compliance with governmental regulations (i. e. Council Directive 96/62/EC of the EU), to monitor ozone concentrations and climate change, or for surveillance of industrial facilities for leakages of pollutants to just name a few applications.

In many of these scenarios it is feasible to have a static sensor network. Therefore, it is not surprising that research on optimal sensor placement at fixed locations exists (e. g. Osborne, Roberts, et al. 2008; Guestrin, Krause, and Singh 2005; Wang et al. 2010). However, better results might be obtainable using mobile robots which can move to interesting areas and acquire more precise data there. Moreover, in some scenarios like disaster response, where timely information is needed, it might not be possible to first deploy an extensive sensor network. In this case mobile robots allow here to quickly identify the interesting regions. The problem of autonomously choosing the best locations for data acquisition is known as *active learning*. In the setting of environmental surveillance Marchant and Ramos (2012) also used the term *intelligent environmental monitoring* (IEM).

In this work, I focus on a scenario proposed as part of the CompLACS project in De Nardi (2013): One or more sources emit a gaseous substance or aerosol which is dispersed by a constant wind. The resulting plume distribution has to be estimated with autonomously controlled unmanned aerial vehicles (UAV). The rather steep concentration gradients and small spatial extent orthogonal to the main dispersion axis add to the difficulty of this problem. With a few UAVs it is not possible to cover the whole volume of investigation using a regular pattern densely in a timely manner. It is necessary to focus on measurements in specific areas. Furthermore, measurement noise has to be taken into consideration.

In previous works swarms of robots have been used to localize the source of a plume (Jatmiko, Sekiyama, and Fukuda 2007; Zarzhitsky, D. F. Spears, and W. M. Spears 2005). These approaches, however, do not allow the usage of only one robot and do not provide one with an estimation of the overall plume distribution. Such an estimation might be important for various reasons such as determining areas in which a threshold is exceeded or a contamination occurred. As Reggente and Lilienthal (2009) noted, though one could try to model the actual fluid dynamics to obtain this information, such computational fluid dynamics models become intractable for real world applications with inaccurate data. Instead they propose to build a statistical model with the gas concentration measurements as random variables.

A widely used statistical model for spatial or spatio-temporal data are *Gaussian* *processes*[1]. In several works (e. g. Stachniss et al. 2008; Marchant and Ramos 2012) the modeled data were actually gas concentrations. Also, there exists some prior work on actively selecting the sampling locations. Stranders, Rogers, and Jennings (2008) do this for discrete locations; Singh et al. (2010) and Marchant and Ramos (2012) for continuous locations. However, none of these approaches is optimal for plume dispersions. This work will porpose and evaluate the improved PDUCB method for the given task.

The thesis is organized as follows. First, a description of the plume modeling scenarios will be given establishing the background of this work. Chapter 3 will give a general introduction into Gaussian Processes and discusses some topics specifically related to the modeling of plume distributions including online updates and active learning. Following in Chapter 4 a number of error measures will be introduced needed to evaluate different approaches. Some further details on how the algorithms were implemented are given in Chapter 5. The results of a number of simulation experiments are presented in Chapter 6, before providing a short outlook on modeling time-varying plume distributions in Chapter 7. Finally, a conclusion will be given.

[1]In geospatial statistics the modeling with Gaussian processes is also known as *kriging*.

# 2. The QRSim Plume Modelling Scenarios

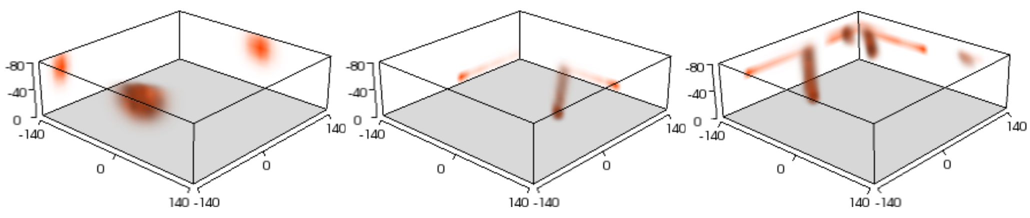
The general plume modeling scenario as tackeled in this thesis is part of the QRSim quadrotors simulator (De Nardi 2013). Several task variations were proposed from which I chose a selection and to which I added some modifications of my own. The task scenarios can be classified along four dimensions: type of dispersion (G, D), presence of sensor noise (NF, SN), single or multiple pollutant sources (SS, MS), single or multiple vehicles (SV, MV).

As long as not otherwise noted location vectors are in the NED (north, east, down) reference frame. Hence, the height of a location $\boldsymbol{x} = (x_1, x_2, x_3)^\top$ is given by $-x_3$.

The most simple scenario is a Gaussian (G) plume without wind as shown in Figure 2.1(a). The pollutant is emitted at a constant rate resulting in a three-dimensional (potentially non-isotropic) Gaussian plume distribution. Given a source location $\boldsymbol{s}$, covariance matrix $\boldsymbol{\Sigma}$, and emission rate $Q$ in g/s the concentration $c(\boldsymbol{x})$ at location $\boldsymbol{x}$ is given as

$$c(\boldsymbol{x}) = Q \cdot \mathrm{s/m^3} \cdot \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{s})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{s})\right). \tag{2.1}$$

A Gaussian dispersion (D) as shown in Figure 2.1(b) is obtained when considering a constant wind parallel to the ground with velocity $u$ measured 6 m above the ground. The plume will be dispersed and form a cone-like distribution along the wind direction. Making a few more assumptions (constant $Q$, steady-state, isotropic diffusion, no ground penetration, and neglegible variation in topography)



(a) Single source Gaussian (b) Single source dispersion (c) Multiple source dispersion

Figure 2.1.: Visualization of different plume dispersions. The rear boundaries of the volume show two-dimensional projections of concentration maxima in the respective directions. Axes scale is in meters.

the analytic expression

$$c(\boldsymbol{x}') = \frac{Q}{2\pi u a \left(x'_1 - s'_1\right)^b} \exp\left(-\frac{\left(x'_2 - s'_2\right)^2}{2a\left(x'_1 - s'_1\right)^b}\right) \left[\exp\left(-\frac{\left(x'_3 - s'_3\right)^2}{2a\left(x'_1 - s'_1\right)^b}\right) + \exp\left(-\frac{\left(x'_3 + s'_3\right)^2}{2a\left(x'_1 - s'_1\right)^b}\right)\right] \quad (2.2)$$

can be derived for the concentration (Stockie 2011). Note that the coordinates $\boldsymbol{x}'$ and $\boldsymbol{s}'$ are expressed in the wind frame of reference. In the dispersion scenarios the wind speed is set to $u = 3\,\mathrm{m/s}$ and the diffusion parameters to $a = 0.33\,\mathrm{m}^{2\,-\,b}$ and $b = 0.86$. The emission rate $Q$ is randomly chosen from a uniform distribution over the interval $0.1\,\mathrm{g/s}$ to $2.5\,\mathrm{g/s}$.

Sensor noise (SN) of the plume sensor is assumed to be additive and distributed according to $\mathcal{N}(0, \sigma_{\mathrm{sn}}^2)$. In the noise free (NF) scenarios no noise was added to the measurements. The standard deviation $\sigma_{\mathrm{sn}}$ was set to $10^{-5}\,\mathrm{g/m^3}$ in the scenarios including noise. The QRSim default scenarios set it to $10^{-2}\,\mathrm{g/m^3}$, but given the low default plume concentration this would require roughly an averaging of 385 samples from one single location to reduce the magnitude of the noise below the magnitude of the concentration values (see Appendix A). Hence, the default scenario is not solvable in a feasible amount of simulation time.

The overall concentration $c(\boldsymbol{x})$ for $n$ sources like in Figure 2.1(c) is obtained by summing the individual contributions $c_i(\boldsymbol{x})$ of each source:

$$c(\boldsymbol{x}) = \sum_{i=1}^{n} c_i(\boldsymbol{x}) \quad (2.3)$$

In the scenarios with multiple sources (MS) $n$ is chosen uniformly out of the range from 1 to 5. With a single source (SS) $n = 1$ is fixed. In both cases the source locations will be randomly chosen from a uniform distribution over the simulated volume.

The starting locations of the UAVs are also chosen randomly and uniformly in the simulated area, but the height is initially set to $x_3 = -10\,\mathrm{m}$. Either a single UAV (SV) or three UAVs (MV) were used.

This gives a number of possible scenarios from which I focussed on the following in this work:

- In Chapter 6.2 I discuss the noise free, single vehicle scenarios G-NF-SS-SV, D-NF-SS-SV, and D-NF-MS-SV. The first is equivalent to the scenario 3A in De Nardi (2013).

- In Chapter 6.3 I consider the dispersion scenarios with noise D-SN-SS-SV and D-SN-MS-SV. Except for the amount of noise these correspond to scenarios 3B and 3C in De Nardi (2013).

- Finally, I will take a look at the usage of multiple vehicles with the scenario D-SN-MS-MV corresponding to scenario 3D in De Nardi (2013).

# 3. Gaussian Processes

A vast number of regression methods has been proposed in the machine learning literature. In this work I use Gaussian Processes as these have been successfully used in a number of studies related to spatial and environmental monitoring including the modeling of gas distributions (e. g. Stranders, Rogers, and Jennings 2008; Marchant and Ramos 2012; Stachniss et al. 2008). Gaussian Processes exhibit a number of desirable features. They are non-parametric and non-linear. Therefore, they do not require any assumptions about the class of underlying functions or limitations of the search space. Also, they provide an estimate of the predictive uncertainties which can be used for a natural exploration-exploitation trade-off.

In the remainder of this chapter the essentials of Gaussian Process regression will be discussed. A more thorough introduction can be found in Rasmussen and Williams (2006).

Let $X = \{\boldsymbol{x}_i | i = 1, \dots, n\}$ be a set of training inputs and $\boldsymbol{y} = (y_1, \dots, y_n)^\top$ a vector of targets. The individual targets are assumed to follow $y_i = f(\boldsymbol{x}_i) + \eta$ with additive noise $\eta \sim \mathcal{N}(0, \sigma_\mathrm{n}^2)$. The complete set of training data will be denoted with $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, \dots, n\}$. We want to learn the function $f(\boldsymbol{x})$ from this training data.

A *Gaussian Process*

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \tag{3.1}$$

imposes a multivariate Gaussian distribution on the space of functions $f(\boldsymbol{x})$. It is completely specified by the mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. Usually, though not necessarily, the mean function is taken to be zero. In most scenarios the choice of the covariance function is much more interesting as it controls features like the smoothness of the predicted underlying function. I will discuss this topic regarding the modeling problem on hand in Section 3.3.

We can now formulate the joint Gaussian prior distribution of the observed training targets and the predicted values $\boldsymbol{f}_*$ at unseen locations $X_*$ (assuming $m(\boldsymbol{x}) = 0$):

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(X,X) + \sigma_\mathrm{n}^2 \boldsymbol{I} & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{3.2}$$

Here $K(X, X')$ are matrices with the elements $(i, j)$ being the covariances $k(\boldsymbol{x}_i, \boldsymbol{x}'_j)$ evaluated for all pairs $\boldsymbol{x}_i \in X$ and $\boldsymbol{x}'_j \in X'$. In the following $\tilde{\boldsymbol{K}} = K(X, X) + \sigma_\mathrm{n}^2 \boldsymbol{I}$ will be used as a shorter notation. By conditioning on the observations the

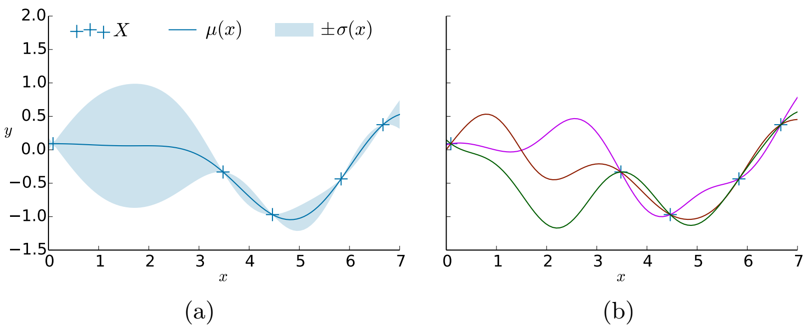


Figure 3.1.: Example of a one-dimensional Gaussian process (using the squared exponential covariance function with a length scale of 1, $\sigma_\mathrm{n}^2 = 0$) conditioned on five training points $X$: (a) shows the mean $\mu(x)$ and predictive standard deviation $\sigma(X_*)$; (b) shows three functions sampled from the process.

predictive distribution for $\boldsymbol{f}_*$ is obtained as

$$\boldsymbol{f}_* | X, \boldsymbol{y}, X_* \sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{cov}(\boldsymbol{f}_*)), \text{ with} \tag{3.3}$$

$$\bar{\boldsymbol{f}}_* = \mu(X_*) = K(X_*, X)\tilde{\boldsymbol{K}}^{-1}\boldsymbol{y}, \tag{3.4}$$

$$\mathrm{cov}(\boldsymbol{f}_*) = K(X_*, X_*) - K(X_*, X)\tilde{\boldsymbol{K}}^{-1}K(X, X_*) \tag{3.5}$$

$$\sigma^2(X_*) = \mathrm{diag}(\mathrm{cov}(\boldsymbol{f}_*)). \tag{3.6}$$

See Figure 3.1 for a visualization of an example Gaussian process.

Even though $\tilde{\boldsymbol{K}}$ is a symmetric, positive-definite matrix it can be ill-conditioned[1] and lead to numerical instabilities. This happens especially for close-by input data points as they occur in a sequential scenario like the plume modeling here.

There are two commonly implemented approaches to counteract the problem of ill-conditioning (cp. Sacks et al. 1989; Neal 1997; Booker et al. 1999; Gramacy and Lee 2008). Firstly, instead of using a general matrix inversion algorithm one can utilize the symmetry and positive-definiteness of $\tilde{\boldsymbol{K}}$ by doing a Cholesky decomposition. This yields a lower, triangular matrix $\boldsymbol{L}$ satisfying $\tilde{\boldsymbol{K}} = \boldsymbol{L}\boldsymbol{L}^\top$. The inverse can then be calculated as $\tilde{\boldsymbol{K}}^{-1} = (\boldsymbol{L}^{-1})^\top \boldsymbol{L}^{-1}$. Secondly, a well conditioned $\tilde{\boldsymbol{K}}$ can be ensured by adding a nugget $g > 0$ (also known as jitter) to the diagonal of the covariance matrix. This will increase all eigenvalues by the same value and thus improve the condition. The addition of a nugget can also be seen as increasing the noise variance $\sigma_\mathrm{n}^2$ and thus allowing the Gaussian Process to match the target less precisely and to become smoother.

[1]The condition $\kappa(\boldsymbol{A})$ of a matrix $\boldsymbol{A}$ is defined as $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$. Using the $L^2$-norm this corresponds to $\kappa(\boldsymbol{A}) = \lambda_1/\lambda_n$, the ratio of the largest eigenvalue $\lambda_1$ and the smallest one $\lambda_n$. If the condition number $\kappa(\boldsymbol{A})$ is too large, the matrix is near-singular and ill-conditioned.

## 3.1. Online Updates

A naive implementation requires a $O\big((n+N)^3\big)$ matrix inversion whenever new data points are added to the Gaussian Process with $N$ being the total number of data points collected so far and $n$ being the number of new data points. However, it is possible to do online updates where only an $n \times n$ matrix has to be inverted. This reduces the complexity of the matrix inversion to $O(n^3)$ and the overall complexity including the necessary matrix multiplications to $O(n \max\{n^2, N^2\})$.

Let us denote the set of inputs already trained on with $X$ and the set of inputs to add as $X'$. The block covariance matrix after adding these new inputs will be

$$\tilde{\boldsymbol{K}}' = \begin{bmatrix} \tilde{\boldsymbol{K}} & K(X, X') \\ K(X', X) & K(X', X') + \sigma_{\mathrm{n}}^2 \boldsymbol{I} \end{bmatrix}. \tag{3.7}$$

The Cholesky factorization can also be written with block matrices

$$\tilde{\boldsymbol{K}}' = \boldsymbol{L}'\boldsymbol{L}'^{\top} = \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{A} & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}^{\top} & \boldsymbol{A}^{\top} \\ \boldsymbol{0} & \boldsymbol{B}^{\top} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}\boldsymbol{L}^{\top} & \boldsymbol{L}\boldsymbol{A}^{\top} \\ \boldsymbol{A}\boldsymbol{L}^{\top} & \boldsymbol{A}\boldsymbol{A}^{\top} + \boldsymbol{B}\boldsymbol{B}^{\top} \end{bmatrix} \tag{3.8}$$

and comparison with equation (3.7) gives the following relations:

$$\boldsymbol{A} = K(X', X) \left(\boldsymbol{L}^{\top}\right)^{-1} \tag{3.9}$$

$$\boldsymbol{B}\boldsymbol{B}^{\top} = K(X', X') + \sigma_{\mathrm{n}}^2 \mathbf{I} - K(X', X)\tilde{\boldsymbol{K}}^{-1} K(X, X') \tag{3.10}$$

As $\boldsymbol{B}\boldsymbol{B}^{\top}$ is symmetric, positive-definite it is possible to obtain $\boldsymbol{B}$ also by a Cholesky decomposition. With the inverse of a block matrix (Petersen and Pedersen 2008, p. 45) we obtain the following relation for the inverse of the updated Cholesky factor:

$$\boldsymbol{L}'^{-1} = \begin{bmatrix} \boldsymbol{L}^{-1} & \boldsymbol{0} \\ -\boldsymbol{B}^{-1} K(X', X) \tilde{\boldsymbol{K}}^{-1} & \boldsymbol{B}^{-1} \end{bmatrix} \tag{3.11}$$

## 3.2. Sparse Approximations

Despite online updates there is a quadratic increase in the complexity with adding more training data points to a Gaussian process. This efficiency problem can be alleviated by using a sparse approximation. A number of different methods has been proposed (chapter 8 in Rasmussen and Williams 2006; Quiñonero-Candela and Rasmussen 2005). Unfortunately, these methods usually assume that all training data is already accessible which is not the case in an online scenario. An exception to this is the online approximation by Csató and Opper (2002).

They replace the inverse covariance matrix $\tilde{\boldsymbol{K}}_t^{-1}$ in Equation 3.5 after $t$ updates with $-\boldsymbol{C}_t$. Normal (full) online updates are performed as discussed in the previous section by updating $-\boldsymbol{C}_t$ respectively the Cholesky factor $\boldsymbol{L}_t^{-1}$.[2] If, however, the "novelty"

$$\tilde{\sigma}_{t+1}^2 = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma_\mathrm{n}^2 - K\big(\{\boldsymbol{x}_*\}, X_t\big)\, \tilde{\boldsymbol{K}}_t^{-1} K\big(X_t, \{\boldsymbol{x}_*\}\big) \tag{3.12}$$

of a new input $\boldsymbol{x}_*$ is below a threshold $\epsilon_\mathrm{tol}$, a reduced update will be performed. The reduced update leaves the size of $\boldsymbol{C}_t$ unchanged.

All inputs used for a full update are called basis vectors and constitute the set $\mathcal{BV}$. The size of this set can be limited by deleting the basis vector with the smallest error whenever the limit is exceeded. The error associated with the $i$-th basis vector is given by

$$e_i = \frac{\big|\big(\tilde{\boldsymbol{K}}_{t+1}^{-1}\boldsymbol{y}\big)_i\big|}{\big(\tilde{\boldsymbol{K}}_{t+1}^{-1}\big)_{ii}}. \tag{3.13}$$

The basis vector deletion requires downdating the approximate covariance matrix $-\boldsymbol{C}_{t+1}$ and performing a reduced updated with the deleted basis vector. Unfortunately, as shown in Appendix B the calculation of $-\boldsymbol{C}_t$ is based on the Cholesky factors and downdating based on Cholesky factorization or a matrix obtained from the factorization is known to be numerical unstable (Björck, Park, and Eldén 1994), especially when the correlation of the training inputs is high. Using other factorizations like the QR factorization a more stable downdating algorithm can be obtained. However, this comes at the cost of the initial factorization being numerical more unstable. Also, it would be possible to directly downdate $\tilde{\boldsymbol{K}}_{t+1}$ and recalculate the Cholesky factorization. Yet, this leads to cubic instead of quadratic complexity for downdating.

It turned out that these issues with numerical stability do not make this sparse online approximation a viable option for plume distribution estimation. The collected samples are highly correlated and make downdating the Cholesky factor too unstable, while using a more stable method impairs efficiency below the non-sparse Gaussian process level. By resigning from deleting basis vector and only using reduced updates based on the novelty $\tilde{\sigma}_{t+1}^2$ not much is gained. The novelty only considers the spatial relation of inputs, but not the actual error in prediction at those locations. Hence, one would either still use almost all data for full updates or use not enough full updates in areas of high concentration where a close sampling is necessary. Luckily, the performance of the non-sparse Gaussian processes was sufficient as at most only a few thousand training inputs were used.

## 3.3. Covariance Functions

The choice of the covariance function determines the assumptions about the most probable functions learned with a Gaussian process. Hence, it is important for reaching the best performance. In this chapter, I will discuss some widely used

[2]The original paper formulates the update a bit different. The equivalence for full updates is shown in Appendix B.

covariance functions and considerations to take into account. A more thorough discussion including further covariance functions is to be found in Rasmussen and Williams (2006, Chapter 4) on which this section is based.

A valid covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel satisfying positive-definiteness

$$\int f(\boldsymbol{x}) k(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}' \geq 0. \tag{3.14}$$

This ensures that the kernel's Gram matrix for a set of inputs $\{x_i | i = 1, \dots, n\}$ with entries $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is also positive-definite and therefore a valid, invertible covariance matrix.

The *smoothness* of the Gaussian process is determined by the covariance function. This is formalized in the notion of how many times the process $f(\boldsymbol{x})$ is *mean square* (MS) *differentiable.* It is differentiable if the mean square limit denoted by mslim in the mean square derivative exists. The MS derivative is given by

$$\frac{\partial f(\boldsymbol{x})}{\partial x_i} = \operatorname*{mslim}_{h \to 0} \frac{f(\boldsymbol{x} + h\mathbf{e}_i) - f(\boldsymbol{x})}{h} \tag{3.15}$$

for the $i$-th direction with the unit vector $\mathbf{e}_i$.

## 3.3.1. Stationary Covariance Functions

A kernel which is only a function of $\boldsymbol{x} - \boldsymbol{x}'$ is called *stationary* and is invariant to translations. Furthermore, it is *isotropic* if it is a radial basis function (RBF) $k(r)$ with $r = |\boldsymbol{x} - \boldsymbol{x}'|$. An isotropic kernel is invariant to all rigid motions.

For stationary kernels the smoothness properties of the resulting Gaussian process can be easily obtained: It is $k$-times MS differentiable if at $\boldsymbol{x} = \mathbf{0}$ the $2k$-th order partial derivatives $\partial^{2k} k(\boldsymbol{x}) / \partial x_{i_1}^2 \dots \partial x_{i_k}^2$ exist and are finite. Thus, the process smoothness is essentially determined by the kernel properties around $\mathbf{0}$.

A common default choice is the *squared exponential* (SE) kernel defined as

$$k_{\mathrm{SE}}(r) = \sigma_{\mathrm{k}}^2 \exp\left( -\frac{r^2}{-2\ell^2} \right) \tag{3.16}$$

with the desired process variance $\sigma_{\mathrm{k}}^2$ and length scale $\ell$. It produces infinitely MS differentiable Gaussian processes. This can, actually, be too smooth in many applications.

The *Matérn class* of covariance functions allows to control the smoothness with a parameter $\nu$. Using the modified Bessel function $K_\nu$ it is given by

$$k_\nu(r) = \sigma_{\mathrm{k}}^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{r\sqrt{2\nu}}{\ell} \right)^\nu K_\nu \left( \frac{r\sqrt{2\nu}}{\ell} \right). \tag{3.17}$$

The resulting Gaussian process will be $k$ times MS differentiable for $k < \nu$. The

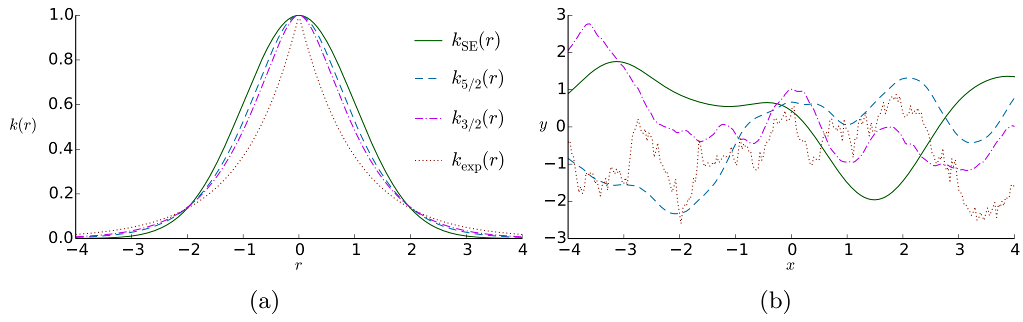


Figure 3.2.: Comparison of covariance functions. (a) Plot of stationary covariance functions with $\sigma_k^2 = 1, \ell = 1$. (b) One function sampled for each of the covariance functions from a Gaussian Process prior with $m(\boldsymbol{x}) = 0$.

parameter $\ell$ denotes again the characteristic length scale.

Typically, only the kernels with $2\nu \in \{1, 2, 3\}$ are used. Usually it is not possible to tell which kernel leads to a better fit for larger $\nu$ from the noisy data. To simplify the function half-integer values are used as the kernel functions will become:

$$k_{5/2}(r) = \sigma_{\mathrm{k}}^2 \left(1 + \frac{r\sqrt{5}}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{r\sqrt{5}}{\ell}\right) \tag{3.18}$$

$$k_{3/2}(r) = \sigma_{\mathrm{k}}^2 \left(1 + \frac{r\sqrt{3}}{\ell}\right) \exp\left(-\frac{r\sqrt{3}}{\ell}\right) \tag{3.19}$$

$$k_{1/2}(r) = k_{\exp}(r) = \sigma_{\mathrm{k}}^2 \exp\left(-\frac{r}{\ell}\right) \tag{3.20}$$

From these kernel functions $k_{\nu=1/2}(r)$ is also known as the *exponential kernel*. Furthermore, note that for $\nu \to \infty$ the squared exponential kernel is recovered. A plot comparing the different covariance functions can be found in Figure 3.2.

## 3.3.2. Non-stationary Covariance Functions

Many phenomena, including the concentrations of gas plumes, are not stationary. Already a Gaussian density function exhibits different optimal length scales. With a long length scale the predicted mean can considerably deviate from the target function as shown for positive $x$ in Figure 3.3. With a short length scale a good fit is obtained. However, the predictive variance along the tail towards negative $x$ is overestimated as the actual rate of change in this area is low.

Non-stationary covariance functions can lessen this problem. Moreover, they allow to model discontinuities at specific places. However, the usage of non-stationary covariance functions for the given plume modeling problem is far from straightforward and might need more prior knowledge than one is willing to assume (in simulations) or effectively has. One would probably have to use different

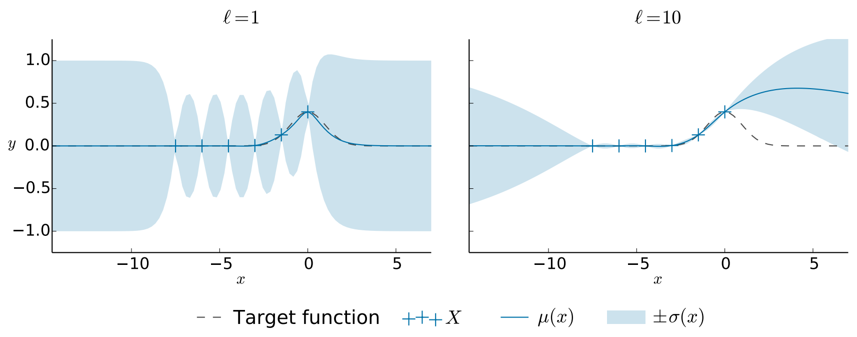


Figure 3.3.: Influence of the kernel length scale on the Gaussian process. In both plots the Matérn kernel with $\nu = 3/2$ was used. On the left a short length scale of $\ell = 1$ was used, whereas a longer length scale of $\ell = 10$ was used on the right.

kernels depending on the scenario (wind/no wind, number of sources) and these would have to be parameterized with the source locations. Otherwise the non-stationarity of the kernel could not relate to the actual non-stationarity of the plume.

Methods for selecting such parameters will be discussed in the next section. Unfortunately, the cost of these methods grows with the number of parameters, which for non-stationary kernels will be larger than for stationary kernels. Matters are complicated even more as it is usually desirable to have a differentiable kernel to be able to use gradient-based optimizers. Moreover, in an active learning scenario there is a limited amount of data in the beginning making the correct estimation of parameters like the source position virtually impossible.

It also has to be taken into account that non-stationary kernels might not be agnostic to the structure of the modeled function. Such a covariance function will lead to better results if the function matches the structural assumptions of the kernel. However, if that is not the case, the results will probably be worse than with stationary kernels agnostic to the structure. Especially when modeling a plume distribution in a real world scenario one would have to consider a multitude of effects like obstacles and local changes in wind. That might make it impossible to derive valid structural assumptions for using a non-stationary kernel.

## 3.4. Hyper-parameter Selection

Though Gaussian processes are non-parametric, the choice of the covariance function will introduce hyper-parameters $\boldsymbol{\theta}$ (i. e. the length scale) which have to be set. In the following I will discuss three methods for doing so.

With the *test set method* all data available $\mathcal{D}$ will be split into two sets $\mathcal{D}_0$ and $\mathcal{D}_*$. The set $\mathcal{D}_0$ is used to train Gaussian processes with different covariance functions and hyper-parameters. For each model the generalization error $E_\mathrm{G}$ over the test set $\mathcal{D}_*$ will be evaluated and the parameters with the minimal generalization error will be chosen. The error measure can be chosen freely with respect to what should be considered a good model. The root mean square error is a typical choice (see also Chapter 4).

If the amount of available data is limited, it is common to use *k-fold cross validation* where $\mathcal{D}$ is split into $k$ disjoint subsets $\mathcal{D}_i$ of equal size and the generalization error will be calculated from $k$ different models using the respective $\mathcal{D}_i$ as test set and the other sets as training data.

The third possibility is to find $\operatorname{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{y}, X, \mathcal{H}_i)$, wherein

$$p(\boldsymbol{\theta}|\boldsymbol{y}, X, \mathcal{H}_i) = \frac{p(\boldsymbol{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\boldsymbol{y}|X, \mathcal{H}_i)} \tag{3.21}$$

consisting of the marginal likelihood $p(\boldsymbol{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)$, a prior $p(\boldsymbol{\theta}|\mathcal{H}_i)$, a normalization factor $p(\boldsymbol{y}|X, \mathcal{H}_i)$, and a set of possible model structures $\mathcal{H}_i$. The normalization factor can be difficult to estimate (Rasmussen and Williams 2006, p. 109). For that reason, even though it can more easily lead to overfitting, often only the marginal likelihood is optimized which is known as *type II maximum likelihood*. For a Gaussian process with $n$ training samples it is given by

$$\ln p(\boldsymbol{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\left(\boldsymbol{y}^\top \tilde{\boldsymbol{K}}^{-1}\boldsymbol{y} + \ln\det\tilde{\boldsymbol{K}} + n\ln 2\pi\right). \tag{3.22}$$

The three summands can be interpreted as the quality of the data fit $\boldsymbol{y}^\top \tilde{\boldsymbol{K}}^{-1}\boldsymbol{y}$, model complexity $\ln\det\tilde{\boldsymbol{K}}$, and a normalization term $n\ln 2\pi$. Hence, the optimization of the marginal likelihood includes an automatic trade-off of model complexity and data fit.

Optimizing the marginal likelihood has the advantage (in comparison to the test set method) that a gradient based optimizer can be used. The partial derivatives of the likelihood are given by

$$\frac{\partial}{\partial\theta_j}\ln p(\boldsymbol{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\left(\left(\tilde{\boldsymbol{K}}^{-1}\boldsymbol{y}\boldsymbol{y}^\top\tilde{\boldsymbol{K}}^{-1} - \tilde{\boldsymbol{K}}^{-1}\right)\frac{\partial\tilde{\boldsymbol{K}}}{\partial\theta_j}\right). \tag{3.23}$$

Nevertheless, all methods require a complete retraining of the Gaussian process as for each update of the hyper-parameters $\tilde{\boldsymbol{K}}^{-1}$ has to be newly calculated. Thus, in an online setting it is far more efficient to keep the hyper-parameters fixed or only update them occasionally.

## 3.5. Active Learning and Bayesian Optimization

A setting in which a learning algorithm can freely choose the next training input is called *active learning*. A general introduction to the topic is provided by Settles (2009). The *Bayesian optimization* (BO) framework provides a method well suited for active learning with Gaussian process. An extensive introduction to Bayesian

**Algorithm 1:** Bayesian optimization algorithm using a Gaussian Process. Taken from Brochu, Cora, and Freitas (2010, p. 6)

```
for $t = 1, 2, \dots$ do
    Find $\boldsymbol{x}_t$ by optimizing the acquisition function over the GP:
    $\boldsymbol{x}_t = \operatorname{argmax}_{\boldsymbol{x}} u(\boldsymbol{x}|\mathcal{D}_{1:t-1})$;
    Sample the objective function: $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$;
    Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\boldsymbol{x}_t, y_t)\}$ and update the GP.
end
```

optimization is given by Brochu, Cora, and Freitas (2010).

In short Bayesian optimization is a method for finding the maximum of an objective function $f(\boldsymbol{x})$. The only condition on $f(\boldsymbol{x})$ is that it is Lipschitz continuous, meaning that a constant $C$ exists, such that

$$\left|f(\boldsymbol{x}) - f(\boldsymbol{x}')\right| \leq C\left|\boldsymbol{x} - \boldsymbol{x}'\right| . \tag{3.24}$$

A closed form expression of $f(\boldsymbol{x})$ may be unknown and it might be costly to evaluate. Because of the latter, Bayesian optimization tries to keep the number of function evaluations small.

The general BO algorithm can be described as follows: Given the prior beliefs about the objective function $p(f)$ a sampling location $\boldsymbol{x}_t$ is chosen. The sample acquired at $\boldsymbol{x}_t$ is used to obtain a posterior according to Bayes' Theorem

$$p(f|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_{1:t}|f)p(f). \tag{3.25}$$

This posterior represents the updated beliefs about the objective function.

The beliefs about the objective function can be expressed using Gaussian processes. Adding further data points corresponds to obtaining an updated posterior. The complete BO algorithm with Gaussian processes is stated in Algorithm 1.

So far it is still open how the sampling location $\boldsymbol{x}_t$ is selected. For that an *utility* or *acquisition*[3] function $u(\boldsymbol{x})$ indicating the expected benefit for choosing $\boldsymbol{x}$ as next training sample is used. Hence, the optimal choice is $\operatorname{argmax}_{\boldsymbol{x}} u(\boldsymbol{x})$. Equivalently, it is possible to use the negative of a loss function $u(\boldsymbol{x}) = -\lambda(\boldsymbol{x})$. In the next section different possible acquisition functions will be discussed in the context of plume distribution modeling.

### 3.5.1. Acquisition Functions

The choice of the acquisition function $u(\boldsymbol{x})$ influences the exploration-exploitation trade-off and on which areas of the input space the learning will be focussed. In the estimation of a plume distribution samples should be acquired mostly at places with high concentrations, but once such an area is well estimated further

[3]The terms will be used interchangeable in the following.

exploration should follow to possibly find further sources.

Marchant and Ramos (2012) proposed the *distance-based upper confidence bound* (DUCB) for a scenario of environmental monitoring where ozone concentrations over US territory were to be measured:

$$u_{\mathrm{DUCB}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + s_{\mathrm{DUCB}}(\boldsymbol{y}) \left[\kappa \cdot \sigma^2(\boldsymbol{x}) + \gamma \cdot d(\boldsymbol{x}, \boldsymbol{x}')\right] \tag{3.26}$$

The mean prediction $\mu(\boldsymbol{x})$ and the predictive variance $\sigma^2(\boldsymbol{x})$ are obtained directly from the Gaussian process, $d(\boldsymbol{x}, \boldsymbol{x}')$ denotes the Euclidean distance of $\boldsymbol{x}$ to the last sample location $\boldsymbol{x}'$. For ease of notation these functions are assumed to be unitless. Precisely this means that $\mu(\boldsymbol{x})$ is divided by $\mathrm{g/m^3}$, $\sigma^2(\boldsymbol{x})$ by $\mathrm{g^2/m^6}$, and $d(\boldsymbol{x}, \boldsymbol{x}')$ by m.

The parameter $\kappa$ controls the exploration-exploitation balance. Higher values give more importance to decreasing the predictive variance and lead to more exploration. The parameter $\gamma \leq 0$ adjusts the distance penalty. Favoring locations near to the current UAV position might decrease the distance travelled and save energy as well as time. The original DUCB formulation by Marchant and Ramos (2012) did not include the scaling factor $s(\boldsymbol{y})$. To match that formulation it has to be set to $s_{\mathrm{DUCB}}(\boldsymbol{y}) = 1$. Another possible choice will be discussed below.

Though, the ozone concentration scenario appears to be similar to the plume modeling problem at hand one should note a certain difference. The ozone concentration is a smooth distribution as Marchant and Ramos (2012) used the squared exponential covariance function to obtain reasonable results. In opposite to that, the spatial distribution of a gas plume is much more localized (this was also noted by Stachniss et al. 2008). Thus, I propose the *plume distance-based upper confidence bound* (PDUCB) acquisition function inspired by DUCB, but adjusted:

$$\begin{aligned} u_{\mathrm{PDUCB}}(\boldsymbol{x}) = (1 - a) \cdot \ln\left(\mu_+(\boldsymbol{x}) + \varepsilon\right) + a \cdot \ln \varepsilon \\ + s_{\mathrm{PDUCB}}(\boldsymbol{y}) \left[\kappa \cdot \left(\sigma^2(\boldsymbol{x}) - \sigma_{\mathrm{n}}^2\right) + \gamma \cdot d^2(\boldsymbol{x}, \boldsymbol{x}')\right] \end{aligned} \tag{3.27}$$

with

$$a = \mathrm{e}^{-\mu_+(\boldsymbol{x})/\varepsilon} \tag{3.28}$$

$$\mu_+(\boldsymbol{x}) = \max\left\{0, \mu(\boldsymbol{x})\right\} \tag{3.29}$$

Using the logarithm of the process mean makes this utility function sensitive for small concentration changes in areas of low concentration. These can hint towards areas with higher concentration. Being sensitive to the same absolute change for high concentrations is not as important. As the concentration might be equal to zero strict positiveness has to be explicitly ensured by a small $\varepsilon > 0$. Also, semi-positiveness of the predictive mean has to be ensured.

It is desirable to have differentiable acquisition functions to be able to use gradient based optimizers. However, due to the logarithm the function would not be differentiable for $\mu(\boldsymbol{x}) \to 0$. Thus, it is weighted with $(1-a)$ and faded out to $\ln\varepsilon$ to restore differentiability (proof in Appendix C).

The noise variance $\sigma_\mathrm{n}^2$ is subtracted from the predictive variance as $\sigma^2(\boldsymbol{x}) \geq \sigma_\mathrm{n}^2$ and this way $\kappa$ needs no adjustment if the noise variance changes.

A further change in PDUCB compared to DUCB is the squaring of the distance which will reduce the penalty around $\boldsymbol{x}'$ (while increasing it further away). This should be advantageous as the unsquared distance function tends to force $\boldsymbol{x}$ much closer to $\boldsymbol{x}'$. Though the next sample should be near to $\boldsymbol{x}'$, it should not be too close to $\boldsymbol{x}'$. Otherwise, not much new information would be gained due to the spatial correlation.

Apart from some explicit values, Marchant and Ramos (2012) do not discuss how to choose the parameters $\kappa$ and $\gamma$. Nevertheless, some observations can be made for both DUCB and PDUCB. Firstly, one should choose $\kappa \cdot \max\sigma^2(\boldsymbol{x}) > \max\mu(\boldsymbol{x})$. Otherwise, one can get stuck in a local maximum as the mean term might get larger than the predictive variance term anywhere in the input space. Even though it can be a good strategy to exploit maxima first, exploration should continue once the distribution around the maximum is accurately known. A too small $\kappa$ is also problematic as it prevents any exploitation. Secondly, $|\gamma|$ should not be too large or the distance penalty would dominate and also lead to one getting stuck in one position $\boldsymbol{x} = \boldsymbol{x}'$. Thirdly, $\varepsilon$ influences the sensitivity for low concentrations and should therefore be small, but large enough to prevent numerical problems in the evaluation of the logarithm. A value of $\varepsilon = 10^{-30}$ seems to work well (see Chapter 6). Finally, it should be noted that PDUCB needs a different scaling $s_\mathrm{PDUCB}(\boldsymbol{y})$ because the logarithm of the mean prediction is used. Assuming $\mu(\boldsymbol{x})$ will be in the range 0 to 1, the range of the logarithmic mean prediction term will be $\ln(1) - \ln(\varepsilon) \approx 70$. Thus, setting $s_\mathrm{PDUCB}(\boldsymbol{y}) = 70$ will make the other parameters of DUCB and PDUCB roughly comparable.

Typically, one knows the spatial dimensions of the input space which allows to estimate a reasonable $\gamma$ in relation to $\kappa$ as the maximal predictive variance is given by $\sigma_\mathrm{n} + \sigma_\mathrm{k}$. However, the maximal $\boldsymbol{y}$ determining $\max\mu(\boldsymbol{x})$ which is important for the absolute values of $\gamma$ and $\kappa$ in relation to the prediction mean might not be known in advance. Thus, it would be helpful to set these parameters automatically from $\boldsymbol{y}$, the data seen so far. This can be done by setting the scaling factor $s(\boldsymbol{y})$ defined for the respective acquisition functions as

$$s_\mathrm{DUCB}(\boldsymbol{y}) = \max\boldsymbol{y} \cdot \mathrm{m}^3/\mathrm{g} \tag{3.30}$$

$$s_\mathrm{PDUCB}(\boldsymbol{y}) = \ln(\max\boldsymbol{y} \cdot \mathrm{m}^3/\mathrm{g} + \varepsilon) - \ln\varepsilon. \tag{3.31}$$

The parameters $\kappa$ and $\gamma$ can then be set independently of the actual values of $\boldsymbol{y}$.

A third potential acquisition function, also balancing exploration and exploitation, can be derived from the work by Osborne, Garnett, and Roberts (2009). They introduce a Bayesian approach for global optimization. Adopting their approach for a one-step look-ahead one first defines a loss function equal to the negative maximum after making a new observation

$$\lambda_{\mathrm{GO}}(y_*) = \begin{cases} -y_* & y_* > \eta \\ -\eta & y_* \leq \eta \end{cases} \tag{3.32}$$

with $\eta = \max \boldsymbol{y}$. From this the expected loss can be determined as

$$\begin{aligned} \Lambda_{\mathrm{GO}}(\boldsymbol{x}) &= \int \lambda_{\mathrm{GO}}(y_*) p(y_* | \boldsymbol{x}, X, \boldsymbol{y}) \, \mathrm{d}y_* \\ &= -\eta - \big(\mu(\boldsymbol{x}) - \eta\big) \, \Phi\Big(\eta; \mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\Big) - \sigma^2(\boldsymbol{x}) N\Big(\eta; \mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\Big) \, . \end{aligned} \tag{3.33}$$

In this equation $N(x; \mu, \sigma^2)$ and $\Phi(x; \mu, \sigma^2)$ are the Gaussian probability density and respectively the Gaussian cumulative distribution function with mean $\mu$ and variance $\sigma^2$. Adding a distance penalty term to the negative of $\Lambda_{\mathrm{GO}}(\boldsymbol{x})$ gives the final utility function

$$\begin{aligned} u_{\mathrm{GO}}(\boldsymbol{x}) &= -\Lambda_{\mathrm{GO}}(\boldsymbol{x}) + \gamma \cdot d^2(\boldsymbol{x}, \boldsymbol{x}') \\ &= \eta + \big(\mu(\boldsymbol{x}) - \eta\big) \, \Phi\Big(\eta; \mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\Big) + \sigma^2(\boldsymbol{x}) N\Big(\eta; \mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\Big) \\ &\quad + \gamma \cdot d^2(\boldsymbol{x}, \boldsymbol{x}'). \end{aligned} \tag{3.34}$$

This approach can be extended to perform a look-ahead of multiple steps. However, this was not done here.

In Figure 3.4 a graphical comparison of the proposed utility functions is given for a one-dimensional example. It can be seen that DUCB heavily focusses on the function maximum without much exploration in other areas. The GO acquisition function also acquires more samples around the maximum, but reduces the uncertainty more equally over the domain. In comparison to these two functions PDUCB seems to also focus around the maxima, but with a wider exploration around those. In Chapter 6.2 I will more closely look on how well these functions work to estimate a plume dispersion with different parameter choices.

### 3.5.2. Multiple UAVs

Using more than one UAV might considerably speed up the estimation of a plume distribution. For that reason I propose a method to extend the target selection with an acquisition function $u(\boldsymbol{x})$ to multiple UAVs. To not impair the performance of the original utility function one mobile robot will be selected as a *master UAV* and uses exactly $u(\boldsymbol{x})$. Without loss of generality we can assign the index $i = 1$ to that UAV. For all other vehicles with $i > 1$ the following modified

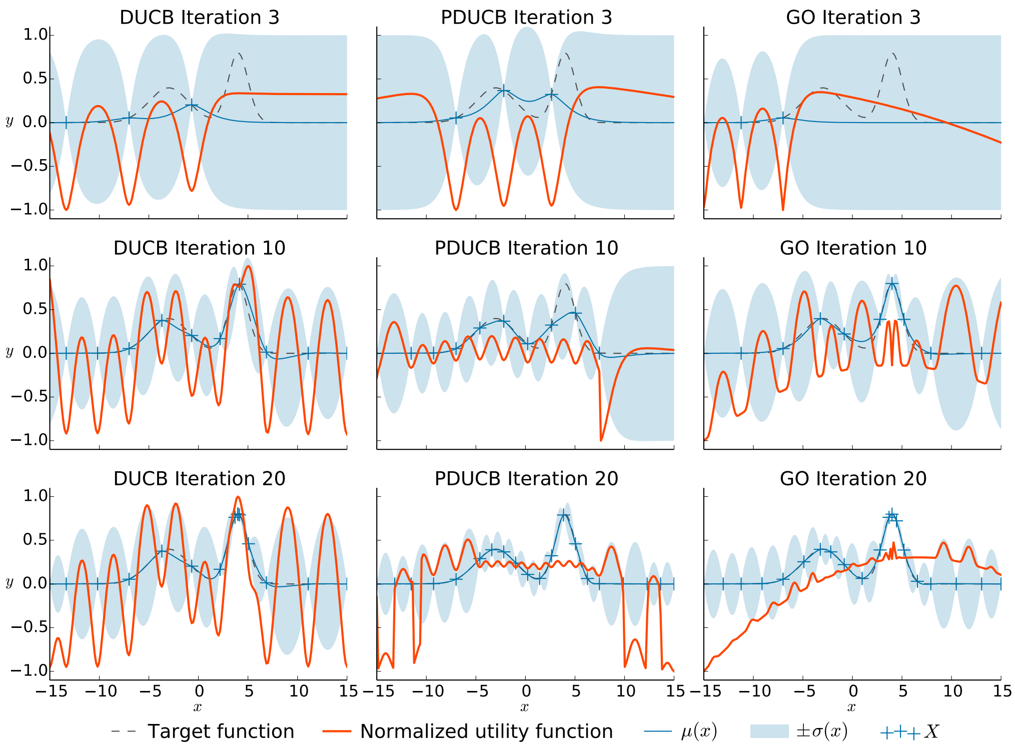


Figure 3.4.: Visualization of acquisition functions. The columns of the plot matrix correspond to the three different acquisition functions (DUCB, PDUCB, GO) and the rows show the state after 3, 10, and 20 iterations. The utility functions were normalized by dividing by $\max_{x\in[-15,15]} |u(x)|$. The parameters used for these plots were $\kappa = 1.25$, $\gamma = -0.0002$, $\varepsilon = 10^{-30}$, $s_{\mathrm{DUCB}}(\boldsymbol{y}) = 1$, and $s_{\mathrm{PDUCB}}(\boldsymbol{y}) = 70$. The initial sample was always chosen at $x_0 = -7$.

acquisition function

$$u_i(\boldsymbol{x}) = u(\boldsymbol{x}) + \rho \sum_{j=1,\ i\neq j}^{n} d^2(\boldsymbol{x}, \boldsymbol{x}_j) \tag{3.35}$$

will be used where $\boldsymbol{x}_j$ is the position of the $j$-th UAV. This modified acquisition function basically introduces a penalty for locations close to other UAVs to spread them out.

### 3.5.3. Initial Search Strategies

As long as no minimal concentration has been measured all of the proposed acquisition functions do not have a unique maximum. Though one could choose sampling locations randomly, this might take a long time until the plume gets discovered. Hence, it is best to employ a more systematic search strategy in the beginning.

Here, three variations will be used. First, surrounding the area at a medium height. Without noise this is sufficient to obtain enough information for a successful usage of the discussed acquisition functions.

Considering noise a second strategy is needed as measurements of low concentrations are not reliably anymore. This strategy, in the following called *complete search*, consists of surrounding the area at different heights until a criterion that a plume as been found is fulfilled. The criterion employed here is that the maximum of all concentration measurements $\max_i y_i$ for one surrounding has to be larger than $5\sigma(y_i)$ with $\sigma(y_i)$ being the standard deviation of the $y_i$.

This complete search may also take a long time before a plume has been found, but should be faster than random exploration. If the wind direction is known (which is not the case in the standard QRSim scenarios), a third strategy called *wind based search* can be used. It improves the second strategy by not doing complete surrounds of the area, but only along the two area edges which are "hit" by the wind. Those are the only two were the plume might be detected.

These bootstrapping strategies can be easily extended to multiple UAVs by assigning different heights to each one.

Most of the samples acquired during this bootstrapping process do not improve the estimation of the plume distribution. Thus, most of them were discarded to improve efficiency. Only the measurements from surrounding the area for the last time before the plume was identified were used for the training of the Gaussian process.

# 4. Error Measures

To be able to compare different statistical models some kind of performance measure, usually in the form of an error measure, is needed. In the plume modeling task one is interested in the deviation of the predicted mean concentrations $\mu(\boldsymbol{x})$ from the true ones $c(\boldsymbol{x})$. When using the $L^2$ norm and integrating over the complete task volume $V$ the *root mean integrated square error* (RMISE)

$$E_{\mathrm{RMISE}} = \sqrt{\frac{1}{v} \int_V \big(c(\boldsymbol{x}) - \mu(\boldsymbol{x})\big)^2 \,\mathrm{d}\boldsymbol{x}} \tag{4.1}$$

with

$$v = \int_V \mathrm{d}\boldsymbol{x} \tag{4.2}$$

is obtained. However, it can be assumed that accurate predictions are more important where the concentration is actually high (cp. Marchant and Ramos 2012). Thus, it might be beneficial to introduce a weighting factor $w(\boldsymbol{x})$ to give the *weighted root mean integrated square error* (WRMISE)

$$E_{\mathrm{WRMISE}} = \sqrt{\frac{1}{v} \int_V \big(c(\boldsymbol{x}) - \mu(\boldsymbol{x})\big)^2 \, w(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}, \tag{4.3}$$

with

$$w(\boldsymbol{x}) = \frac{c(\boldsymbol{x}) - \min_{\boldsymbol{x}'} c(\boldsymbol{x}')}{\max_{\boldsymbol{x}'} c(\boldsymbol{x}') - \min_{\boldsymbol{x}'} c(\boldsymbol{x}')}. \tag{4.4}$$

Note that in areas with concentration of almost or even exactly zero the WRMISE will always be close to zero and thus allowing the model to make highly inaccurate predictions. Therefore, the WRMISE should not be used as sole measure in plume modeling. Using both errors it is possible to ensure good overall fit of the prediction without large inaccuracies and to compare which model has the better fit in the interesting areas.

Unfortunately, the RMISE and WRMISE cannot easily be calculated analytically and one has to restrain to approximating the integral from a finite set $\{\boldsymbol{x}_i | i = 1, \ldots, n\}$ of samples. If the $x_i$ are distributed according to the probability density function $p(\boldsymbol{x})$, the approximation has the form

$$\hat{E}_{\text{RMISE}} = \sqrt{\frac{1}{vZ}\sum_{i=1}^{n}\frac{\left(c(\boldsymbol{x}_i)-\mu(\boldsymbol{x}_i)\right)^2}{p(\boldsymbol{x}_i)}} \tag{4.5}$$

$$\hat{E}_{\text{WRMISE}} = \sqrt{\frac{1}{vZ}\sum_{i=1}^{n}\frac{\left(c(\boldsymbol{x}_i)-\mu(\boldsymbol{x}_i)\right)^2 w(\boldsymbol{x}_i)}{p(\boldsymbol{x}_i)}}. \tag{4.6}$$

The normalization constant $Z$ is given by

$$Z = \sum_{i=1}^{n}\frac{1}{p(\boldsymbol{x}_i)}. \tag{4.7}$$

The probability density $p(\boldsymbol{x})$ can be estimated using Gaussian kernel density estimation (KDE). There exist different methods to determine the bandwidth parameter of the KDE. In this work Scott's Rule (Scott 1992) was used.

## 4.1. Selecting Samples for Error Approximation

Up to now it is still open how to select the $\boldsymbol{x}_i$ for approximating the error measures. Using a regularly spaced grid, it is either likely to miss the important areas as the plume is relatively localized, or it is so fine grained that the evaluation of the error takes a long time. Thus, I used a slightly modified Metropolis-Hastings (MH) algorithm to accumulate samples in areas with a high concentration.

The standard Metropolis-Hastings (Chib and Greenberg 1995) is used to sample from a probability distribution $p(\boldsymbol{x})$ given only a function $f(\boldsymbol{x})$ proportional to $p(\boldsymbol{x})$. It starts at a random location $\boldsymbol{x}_1$. Then in each iteration $i > 1$ a new candidate location $\boldsymbol{x}_*$ is picked from a symmetric[1] proposal distribution and an acceptance ratio $\xi = f(\boldsymbol{x}_*)/f(\boldsymbol{x}_i)$ is calculated. The candidate $\boldsymbol{x}_*$ is accepted as $\boldsymbol{x}_{i+1} = \boldsymbol{x}_*$ with probability $\xi$ ($\xi \geq 1$ automatically accepts). If it is rejected, $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i$ will be used.

To select the samples for the error approximation this algorithm can be used with the true concentrations $f(\boldsymbol{x}) = c(\boldsymbol{x})$. For this purpose it is not necessary that the samples exactly follow a specific probability distribution. That makes it possible to make a few adoptions for better results.

First of all, $\boldsymbol{x}_*$ can always be added to the set of samples independent of acceptance or rejection instead of the accepted sample. This makes each location unique (as long as the same location does not get proposed twice). Having multiple instance of the same location within the samples would not increase the accuracy of the error approximation. Also, this leads to a few more samples towards the concentration tails where the concentration is already low but spatially close to

[1] $Q(\boldsymbol{a}|\boldsymbol{b}) = Q(\boldsymbol{b}|\boldsymbol{a})$

high concentrations. Thus, the approximation around the concentration slope can be expected to be better.

A further change concerns the initial location $\boldsymbol{x}_1$. Choosing at random might place it in an area with zero concentration which renders the acceptance ratio $\xi$ undefined. Hence, it is better to base the initial location near a source location. A placement directly at the source location is not possible given that a plume dispersion (Equation 2.2) is undefined that place.

In case of multiple sources the sampling should be started from each source location and the resulting sets should be joined. The concentration between sources can be rather low making it unlikely to switch from one plume to another. Also, the Metropolis-Hastings algorithm is unlikely to sample in low concentration areas, especially in some distance to the plumes. Thus, a number of uniformly sampled locations should be added.

To “smooth” out the samples around the plumes even more one can use only every $k$-th sample of the Metropolis-Hastings algorithm and use the remaining samples as center of Gaussian distributions and draw $k$ more samples from each of these.

## 4.2. QRSim Reward

The plume modeling scenarios in De Nardi (2013) themselves define a reward

$$R = -\sum_{i=1}^{n} \big(c(\boldsymbol{x}) - \mu(\boldsymbol{x})\big)^2 \tag{4.8}$$

as a performance measure. This is essentially the negative of $\hat{E}^2_{\text{RMISE}}$ without normalization for the sampling density. Thus, with a non-uniform sampling this reward will give a biased estimate attributing more importance to areas with more sample locations.

The set of $\boldsymbol{x}_i$ used to calculate the reward in QRSim consists ouf of two parts. One-fifth is sampled uniformly over the whole volume, whereas the remaining samples are taken uniformly from the areas where the concentration exceeds the limit $c_{\min} = Q \cdot 10^{-3} \cdot \mathrm{s/m^3}$.

The bias introduced by that towards the interesting regions is not necessarily bad. The same is done in the WRMISE. Unfortunately, the sampling strategy will not sample at the low side of a (steep) concentration boundary and will give a bad estimate of the reward around those boundaries. In contrast to that, the Metropolis-Hastings based sampling algorithm acquires also some samples in the low concentration area around a high concentration. This difference is visualized in Figure 4.1.

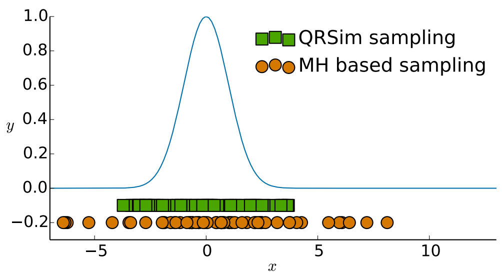


Figure 4.1.: One-dimensional comparison of the Metropolis-Hastings based sampling and the QRSim sampling for error estimation. For both approaches 60 samples with regard to the plotted Gaussian density were taken excluding the uniformly distributed samples. The $y$ position of the scatter marks has no meaning in this plot. For the MH based sampling in this plot every fifth sample was used. A Gaussian was used as proposal distribution with $\sigma = 2$. The same distribution was used to create five additional samples for each MH based sample.

## 4.3. Normalized Error

For a reliable comparison of the models it is not sufficient to compare them based on a single trial. A good model should provide a low error in different trials.

In the plume modeling scenarios the concentration density and spatial extent of the plume varies. The error measures directly depend upon the absolute concentration values and, therefore, also depend on the specific trial.

Directly averaging the error over trials would give a higher weighting to trials with high concentration densities or a large plume extent. A good plume modeling method should adapt to the concentration levels and be more precise for lower overall concentrations. To achieve a fair weighting of the trials the error can be normalized as

$$F = \frac{E}{E_0} \tag{4.9}$$

for each trial, where $E$ is the current error estimate and $E_0$ the error estimate for an all zero prediction. The error estimate can be freely chosen with regard to its properties. Note that this equation can also be applied to the reward $R$ as error measure as the minus cancels out. When averaging $F$ over trials each trial is weighted equally.

The normalized error $F$ has also the advantage of being readily interpretable. For $F > 1$ the prediction is worse than an all zero prediction. This should not happen with a viable modeling method. For $F < 1$ the prediction is better than an all zero prediction and the error has been decreased by $(1 - F) \cdot 100\,\%$.

# 5. Technical Details

To implement the discussed algorithms in a functioning system some further details have to be taken into consideration. I will discuss these in the following after giving a short general overview of the implementation.

## 5.1. Implementation

All simulations were performed with QRSim (De Nardi 2013), a quadrotor simulator developed specifically to test high level tasks. It supports multiple UAVs, which are simulated with realistic platform dynamics. Also, equipped sensors (e. g. GPS, IMU) are simulated with different sources of inaccuracies. This includes wind influencing the vehicles as well as the plume dispersions.

I decided to implement the algorithms (i. e. Gaussian processes, acquisition functions) in Python because of its high-level programming constructs and excellent scientific computing support with NumPy and SciPy (Oliphant 2007). Communication between the Python part and QRSim implemented in MATLAB were done with a TCP interface based on the Google protocol buffers[1].

Though there exist several Gaussian process implementations for Python like for example Scikit-learn (i. e. Pedregosa et al. 2011), none supports online updates to my knowledge. For that reason I developed an own implementation.

## 5.2. Function optimization

For finding the maximum of an acquisition function the SciPy wrapper of the FORTRAN implementation of the L-BFGS-B algorithm (Byrd et al. 1995; Zhu et al. 1997) was used. As gradient based optimizer with the possibility to constrain the search space to the task volume it is well suited for the task.

To choose the starting location $\boldsymbol{x}_0$ the utility function was evaluated on a coarse $5 \times 5 \times 5$ grid and the location with the maximal value was chosen. The convergence parameters were set to $\mathit{pgtol} = 10^{-10}$ and $\mathit{factr} = 100$. The rather flat DUCB gradient required this strict settings. For the other acquisition functions a good convergence was also possible with higher values (i. e. $\mathit{pgtol} = 10^{-5}$ and $\mathit{factr} = 10^7$). Nevertheless, the same parameters were used for all utility functions.

Noisy plume measurements produce a large number of local maxima in the utility function. Hence, in those scenarios the optimization has to be performed multiple times using some additional starting points. For this Gaussian distributions with a

[1]`https://developers.google.com/protocol-buffers/`

standard deviation of 5 m centered on the current UAV position and the location of the maximal recorded plume measurement were used. From each of these Gaussian five additional starting locations for the optimization were sampled.

## 5.3. UAV Control

The QRSim simulator accepts different UAV control commands including way-points and target velocities. Setting directly the way-points would be natural as the optimization of the acquisition function leads to target coordinates. However, the QRSim way-point controller did not proof to be very reliable and UAVs leaving the simulation area were not uncommon. To circumvent this, a different algorithm was used. Each way-point $\boldsymbol{t}_i$ was translated to velocity commands $\boldsymbol{v}_i$ sent to QRSim for the $i$-th out of $n$ UAVs with

$$\boldsymbol{v}_i = \begin{pmatrix} d_{i,1} \min\{1, v_{\max,1}/d_{i,\mathrm{hor}}\} \\ d_{i,2} \min\{1, v_{\max,2}/d_{i,\mathrm{hor}}\} \\ \min\{v_{\max,3}, \max\{-v_{\max,3}, d_{i,3}\}\} \end{pmatrix} \tag{5.1}$$

$$d_{i,\mathrm{hor}} = \sqrt{d_{i,1}^2 + d_{i,2}^2} \tag{5.2}$$

$$\begin{aligned} \boldsymbol{d}_i &= \left(d_{i,1}, d_{i,2}, d_{i,3}\right)^\top \\ &= \operatorname{diag}(\boldsymbol{v}_{\max}) \left( u_1 \left(\boldsymbol{t}_i - \boldsymbol{x}_i\right) + u_2 \sum_{j=1,\ i\neq j}^{n} \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^3} \right) \end{aligned} \tag{5.3}$$

where $\boldsymbol{x}_i$ are the current UAV positions, $u_1 = 0.025/\mathrm{m}$ and $u_2 = 5\,\mathrm{m}^2$ are scaling constants, and $\boldsymbol{v}_{\max} = (v_{\max,1}, v_{\max,2}, v_{\max,3})^\top = (6\,\mathrm{m/s}, 6\,\mathrm{m/s}, 6\,\mathrm{m/s})^\top$ a vector of speed limits[2]. The $\boldsymbol{d}_i$ vector components consist of two terms. One directs the speed towards the target proportional to the remaining distance resulting in a slow down near to the target. The remaining term acts like a repelling force between the UAVs to keep them from colliding. It is proportional to the inverse of the square of the distance. The power of three occurs because the direction vector $\boldsymbol{x}_i - \boldsymbol{x}_j$ has to be normalized. With Equation 5.1 the velocity will be limited while keeping the overall horizontal direction.

To prevent the UAVs from going astray a safety margin of 10 m is defined at the boundaries of the simulated volume. When an UAV enters this margin in one dimension the corresponding velocity component will be set to $\pm\boldsymbol{v}_{\max}$.

A target $\boldsymbol{t}_i$ is considered to be reached when $|\boldsymbol{t}_i - \boldsymbol{x}_i| < 3\,\mathrm{m}$.

[2]QRSim additionally applies its own speed limit independently per direction in the UAV body frame of reference. It is 3 m/s for the two horizontal axes.

# 6. Evaluation and Simulation Experiments

A number of simulation experiments has been performed to evaluate the proposed methods for modeling plume distributions. First, the most suitable covariance function and hyper-parameters were obtained. These results were then used to compare the different acquisition functions based on the noise-free scenarios. Subsequently, the performance of PDUCB was evaluated on the noisy scenarios and using multiple UAVs.

Note that, in all calculations of error measures, predicted concentration values below zero were set to exactly 0 as a negative concentration is physically not possible.

## 6.1. Best Covariance Function for Plume Modelling

To determine the best covariance function including its parameters to approximate a plume distribution the different functions were evaluated using the test set method. For each of the single source Gaussian (G-NF-SS-SV), the single source dispersion (D-NF-SS-SV), and the multiple source dispersion (D-NF-MS-SV), all without noise, 50 random instances were created. For each instance a set of sampling locations was generated using the Metropolis-Hastings based technique described in Chapter 4.1. Herein, every fifth Metropolis-Hastings sample was used in the final set and was used as mean of Gaussian with a standard deviation $\sigma = 6\,\mathrm{m}$ to draw five more samples to include in the final set. The proposal distribution of the Metropolis-Hastings algorithm was also a Gaussian with standard deviation $\sigma = 6\,\mathrm{m}$. In addition, 1000 uniformly samples were added to the final set of samples. All samples outside of the scenario volume were dismissed. From all kept sample points 1000 were randomly selected for training and the rest was used as test set to determine the error.

Obtaining the training samples in this way should roughly mirror a good sampling with an UAV with many samples in the areas of high concentration and a few in the remaining areas. The advantage using this way of sampling is that it allows us to test different kernels independent of the exact behavior of the UAV and time consuming simulation of it.

The kernels tested were the squared exponential, the Matérn kernel with $\nu = 5/2$, the Matérn kernel with $\nu = 3/2$, and the exponential kernel. The length scales tested ranged from $1\,\mathrm{m}$ to $100\,\mathrm{m}$. The process variance was fixed as $\sigma_{\mathrm{k}}^2 = 1$. Note that this parameter has no effect on the predictive mean as long as the assumed noise variance $\sigma_\mathrm{n}^2$ is zero.

The normalized error is plotted in Figure 6.1 for the different kernels and error measures. The minimum is roughly the same for all kernels and lies around $\ell = 5\,\mathrm{m}$. However, the behavior differs considerably for non-optimal length scales. The smoother (the more often the kernel is differentiable) the more the error increases for large length scales. Especially, for the squared exponential covariance function this increase is abrupt. Only for very large length scales it decreases again for this kernel.

Comparing the WRMISE to the RMISE the former one stays low even for larger length scales. This indicates that in the area of the plume (also due to the more dense sampling) a good fit is still obtained, but around that area the prediction gets worse. Thus, the steep concentration gradients around the plume are not well captured in that case.

The results give also an idea about how good of a fit can be expected at best when using an UAV instead of direct sampling. Whereas the normalized error decreases to nearly zero for the single source Gaussian, it stays above 0.6 for the dispersion scenario with the more localized plume distribution. The reward error measure is decreased to lower levels, but this is likely to underestimation of the error at the plume boundaries as argued in Chapter 4.2.

Besides the error measures the log likelihood of each trained Gaussian process was calculated. In Figure 6.2 the average over trials is plotted. Only for the squared exponential kernel the maximum of the log likelihood corresponds to the minimum of the RMISE. Towards longer length scales the likelihood declines very steeply. Using the log likelihood to estimate the length scales for the other covariance functions would largely overestimate it.

Taken these results together it is best to choose a non-smooth kernel with a length scale of $\ell = 5\,\mathrm{m}$. As it is advantageous to be able to use a gradient based optimizer for the optimization of acquisition functions, I decided to use the Matérn kernel with $\nu = 3/2$ in the further experiments, which gives a once mean square differentiable Gaussian process as opposed to the exponential kernel. Unfortunately, optimizing the length scale using the likelihood would not give good results and I fixed the length scale at $\ell = 5\,\mathrm{m}$. Also, including a prior in the log likelihood does not help here. For example to shift the maximum of the likelihood for the chosen kernel to 5 m a Gaussian prior would need a standard deviation of less then $\sigma_\ell < \mathrm{e}^{-2078}/\sqrt{2\pi} \approx 0$ (see Apendix D). Thus, effectively resulting in a fixed length scale.

## 6.2. Comparison of Utility Functions

Given the kernel chosen in the previous section I continued to compare the different utility functions in the noiseless scenarios single source Gaussian (G-NF-SS-SV), single source dispersion (D-NF-SS-SV), and multiple source dispersion (D-NF-MS-SV).

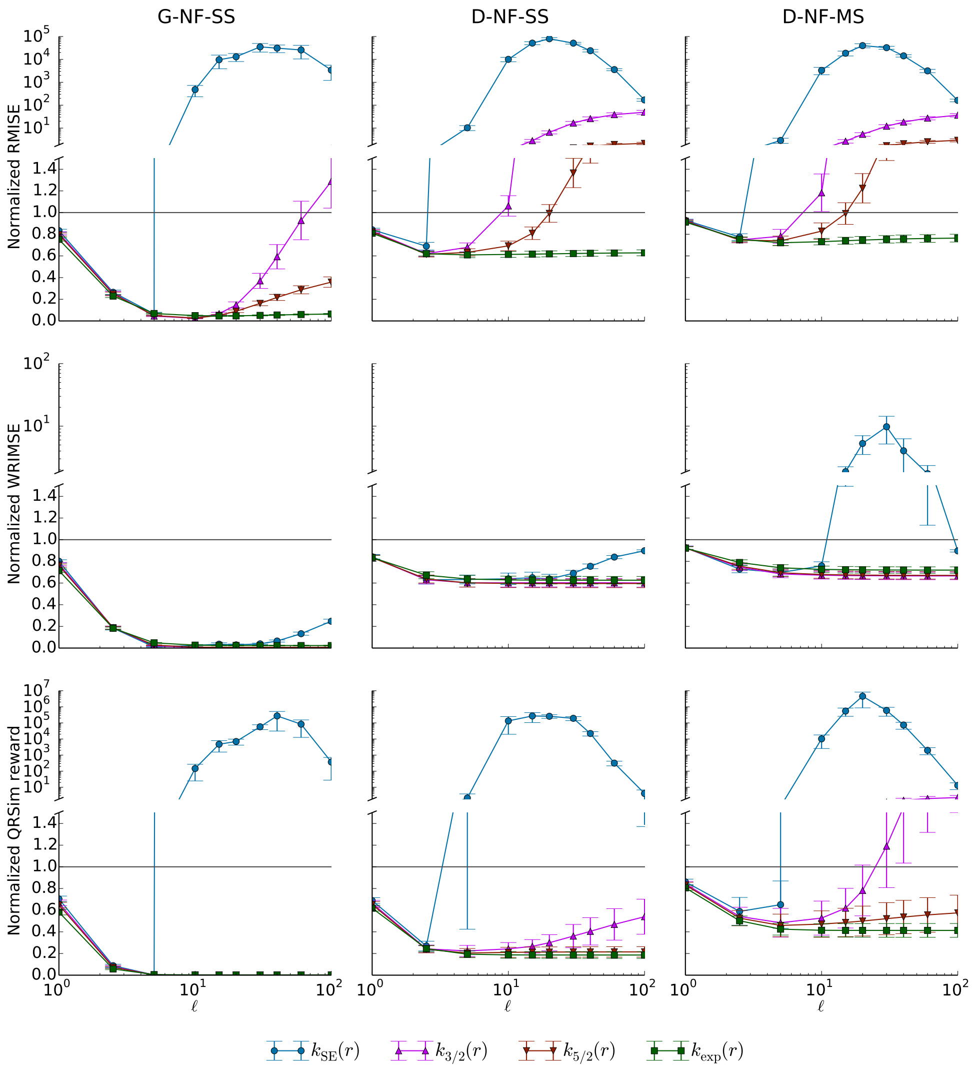


Figure 6.1.: The normalized error for different covariance function in dependence of length scale averaged over trials. The rows correspond to the RMISE, WRMISE, and QRSim reward error measures. The columns correspond to a single source Gaussian (G-NF-SS), a single source Gaussian dispersion (D-NF-SS), and a multiple source Gaussian dispersion (D-NF-MS). All scenarios were simulated without sensor noise. Error bars represent the standard error. The boundary of 1.0 where the error of the trained Gaussian process is larger than an all zero prediction is marked with a horizontal line.

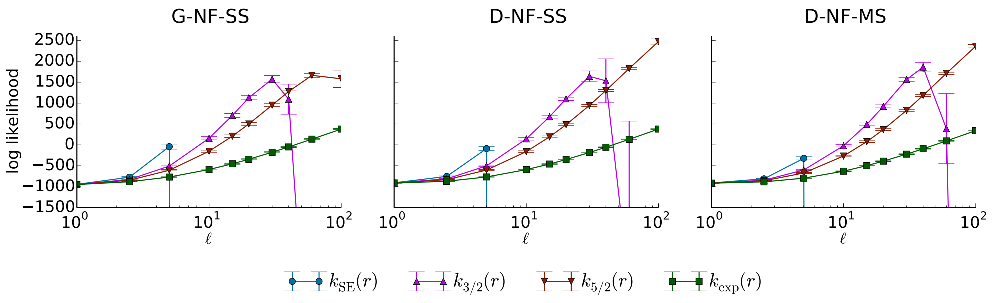


Figure 6.2.: The average log likelihood of the training data in dependence of the length scale using different kernels. Each of the three plots shows one of the single source Gaussian (G-NF-SS), single source Gaussian dispersion (D-NF-SS), and multiple source Gaussian dispersion (D-NF-MS) without sensor noise. Error bars represent the standard error.

For each given scenario 20 trials were performed. In each run the UAV first surrounded the simulation area in a height of 40 m with a margin of 10 m to the boundaries of the simulated volume. After that further way-points were chosen with one of the acquisition functions discussed in Chapter 3.5.1. The optimization of those functions has been described in Chapter 5.2.

Each trial was allowed to run until a maximum of 3000 plume measurements was acquired. A plume measurement was taken every second. When a new target way-point was within 3 m of the previous one, the UAV was considered to become stuck in a maximum of the acquisition function and the simulation was stopped at that point to reduce overall simulation time.

The error measures were estimated as described in Chapter 4. The samples for that were 1000 uniformly distributed sampling locations, every tenth of 4200 locations from the Metropolis-Hastings algorithm with Gaussian proposal distribution with standard deviation $\sigma = 10\,\mathrm{m}$, and 10 more locations sampled from the proposal distribution for each of the included Metropolis-Hastings samples.

I tested all three utility functions proposed in Chapter 3.5.1: DUCB, PDUCB (with $\varepsilon = 10^{-30}$), and GO. DUCB was tested with a constant scaling factor of $s_{\mathrm{DUCB}}(\boldsymbol{y}) = 1$ and the automatic scaling in Equation 3.30. PDUCB was tested with a constant scaling factor of $s_{\mathrm{PDUCB}}(\boldsymbol{y}) = 70$ and the automatic scaling in Equation 3.31. Furthermore, a parameter search over the weighting factors $\kappa \in \{0.1, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ and $\gamma \in \{0\} \cup \left\{-10^p | p = -9, -8, \ldots, -2\right\}$ was performed. Note that for the GO utility function the $\kappa$ parameter has no effect.

Figure 6.3–6.5 visualize the normalized error for the different scenarios. The respective parameters and values of the minima (excluding the QRSim reward) are listed in Table 6.1–6.3. The average reduction (over trials) of the RMISE against simulation time in the single source Gaussian scenario (G-NF-SS-SV) is plotted in Figure 6.6 and looks essentially the same for WRMISE and the QRSim reward and therefore it is not shown for those measures. Finally, Figure 6.7 visualizes an

| Utility function | $\kappa$ | $\gamma$ | RMISE Mean ng/m$^3$ | RMISE SD ng/m$^3$ | Norm. RMISE Mean | Norm. RMISE SD |
|---|---|---|---|---|---|---|
| DUCB | 1.25 | $-10^{-7}$ | 261.27 | 184.79 | 0.29 | 0.20 |
| auto-scaled DUCB | 1.50 | $-10^{-8}$ | 260.06 | 180.16 | 0.31 | 0.25 |
| PDUCB | 0.10 | $-10^{-9}$ | 206.79 | 242.72 | 0.18 | 0.13 |
| auto-scaled PDUCB | 0.10 | $-10^{-7}$ | 204.51 | 213.98 | 0.18 | 0.12 |
| GO | 0.10 | $-10^{-8}$ | 770.91 | 376.38 | 0.80 | 0.13 |

| Utility function | $\kappa$ | $\gamma$ | WRMISE Mean ng/m$^3$ | WRMISE SD ng/m$^3$ | Norm. WRMISE Mean | Norm. WRMISE SD |
|---|---|---|---|---|---|---|
| DUCB | 1.25 | $-10^{-7}$ | 121.46 | 122.79 | 0.21 | 0.22 |
| auto-scaled DUCB | 1.50 | $-10^{-8}$ | 127.55 | 123.92 | 0.25 | 0.28 |
| PDUCB | 0.10 | $-10^{-6}$ | 100.19 | 142.12 | 0.13 | 0.14 |
| auto-scaled PDUCB | 0.10 | $-10^{-7}$ | 100.58 | 137.84 | 0.13 | 0.13 |
| GO | 0.10 | $-10^{-8}$ | 489.22 | 258.44 | 0.80 | 0.15 |

Table 6.1.: The minimal obtained error (RMISE and WRMISE) for each acquisition function and the parameter values used in the single source Gaussian scenario (G-NF-SS-SV).

example UAV trajectory for the PDUCB acquisition function.

This is a rich dataset from which a number of insights can be gained. First of all it can be noted that the normalized QRSim reward attributes almost always a better performance to the algorithms than the other two error measures. This is consistent with the argument in Chapter 4.2 that the negative reward may be underestimated at concentration boundaries. Moreover, there seems to occur some kind of artifact leading to a very low QRSim reward for one parameter setting ($\kappa = 1.25, \gamma = -10^{-6}$) in the multiple source scenario.

Turning to the actual utility functions it can be noted that the GO function does not perform very well. Even in the single source Gaussian scenario the RMISE is only reduced by about 20 % and in the other two scenarios it performs even worse.

Comparing DUCB and PDUCB the latter one consistently performs better with a reduction of the RMISE and WRMISE by at least additional 8 % and in the dispersion scenarios even more. Despite that, the standard deviation of PDUCB is almost always higher than that of DUCB in the dispersion scenarios.

In the single source Gaussian scenario PDUCB proves to be robust against the choice of $\kappa$ as for all values very good results are obtained. This picture is a bit more noisy in the dispersion scenarios. It seems that too low values ($\kappa < 0.5$) degrade performance. This is consistent with the argument in Chapter 3.5.1 that

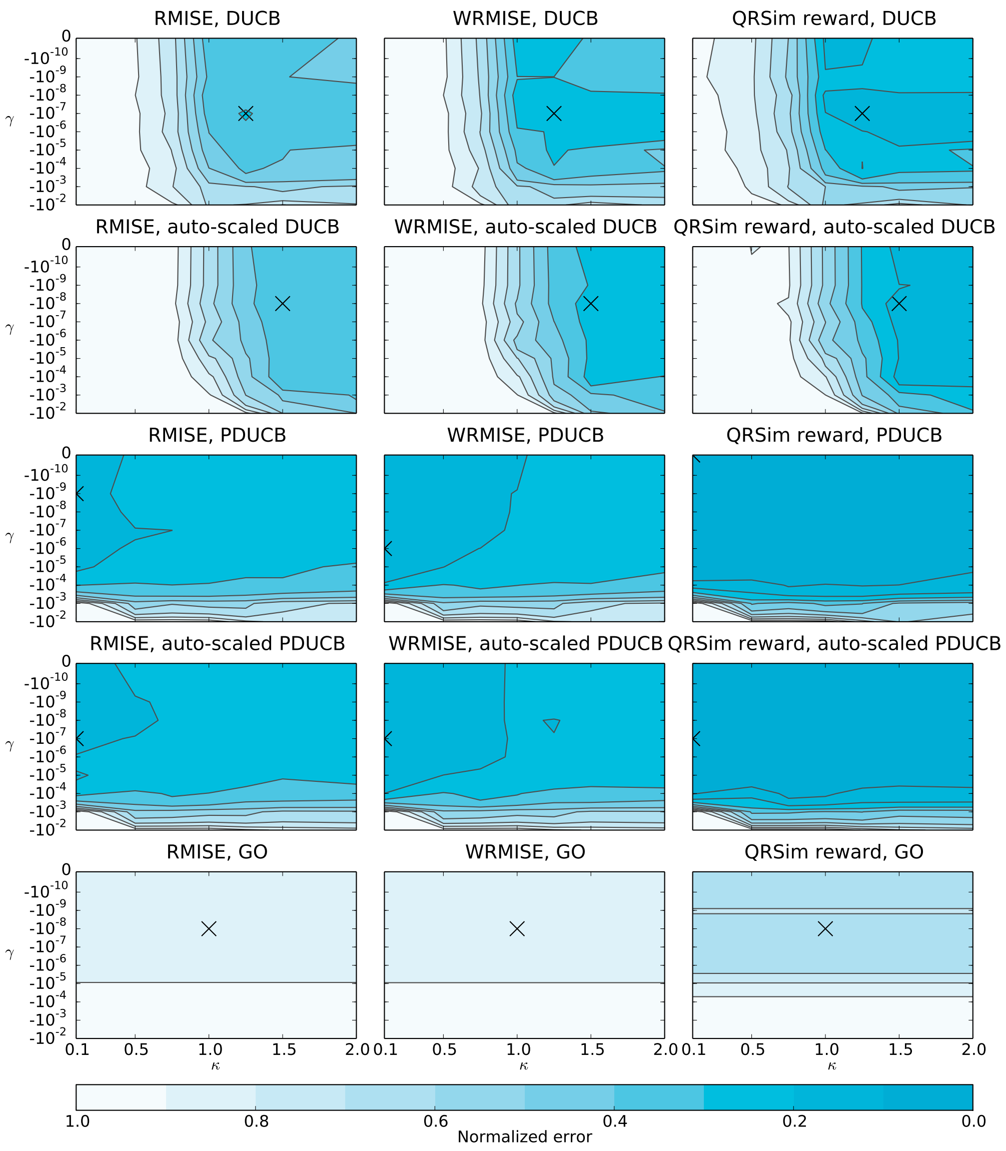


Figure 6.3.: The normalized error for different measures, utility functions, and parameters in the noiseless single source Gaussian scenario (G-NF-SS-SV). The columns represent the RMISE, WRMISE, and QRSim reward error measure. The rows represent the DUCB, auto-scaled DUCB, PDUCB, auto-scaled PDUCB and GO utility functions. The auto-scaled versions use the scaling factor defined in Equations 3.30 and 3.31, in contrast to a constant scaling factor. The minimum of each plot is marked with cross.

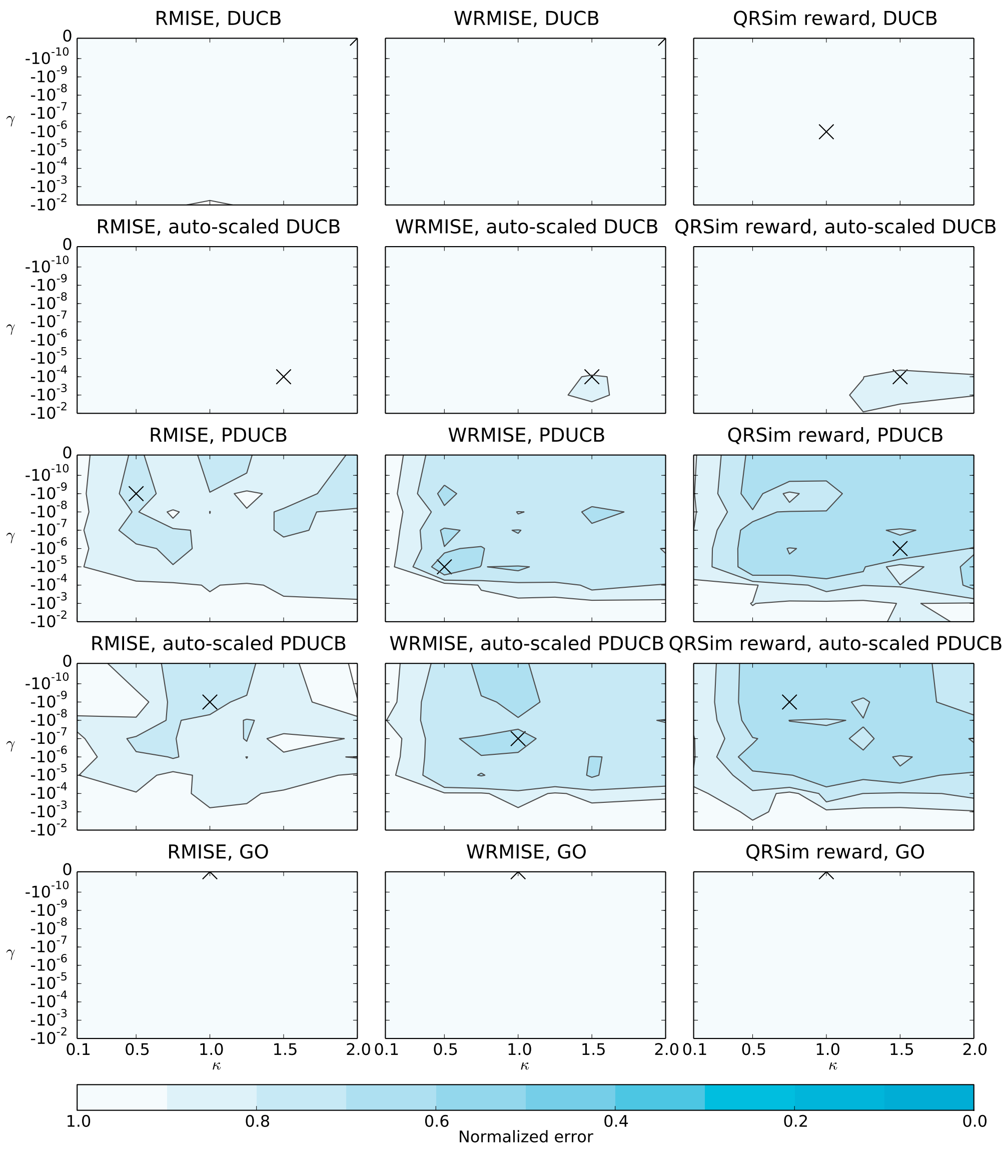


Figure 6.4.: The normalized error for different measures, utility functions, and parameters in the noiseless single source Gaussian dispersion scenario (D-NF-SS-SV). The columns represent the RMISE, WRMISE, and QRSim reward error measure. The rows represent the DUCB, auto-scaled DUCB, PDUCB, auto-scaled PDUCB and GO utility functions. The auto-scaled versions use the scaling factor defined in Equations 3.30 and 3.31, in contrast to a constant scaling factor. The minimum of each plot is marked with cross.

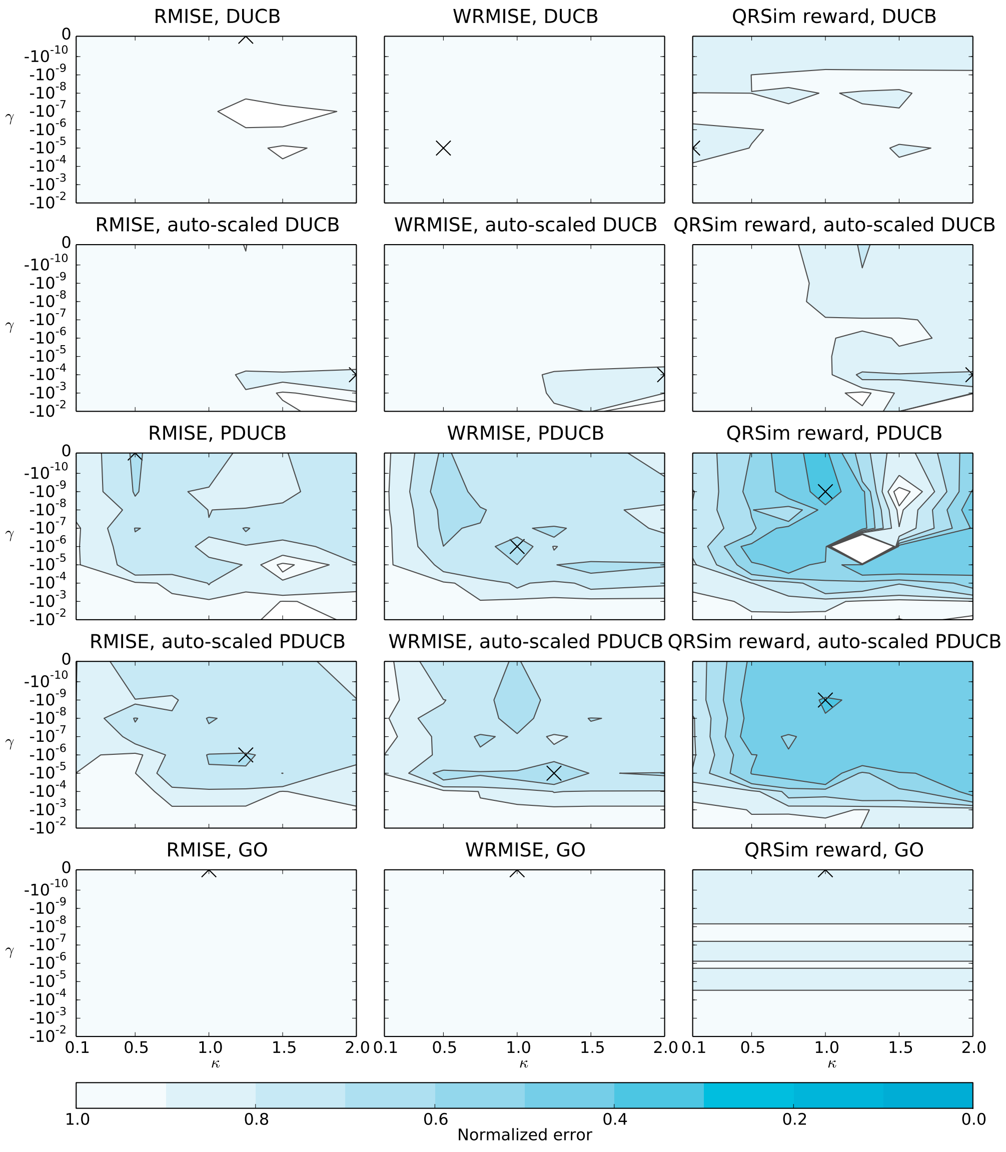


Figure 6.5.: The normalized error for different measures, utility functions, and parameters in the noiseless multiple source Gaussian dispersion scenario (D-NF-SS-SV). The columns represent the RMISE, WRMISE, and QRSim reward error measure. The rows represent the DUCB, auto-scaled DUCB, PDUCB, auto-scaled PDUCB and GO utility functions. The auto-scaled versions use the scaling factor defined in Equations 3.30 and 3.31, in contrast to a constant scaling factor. The minimum of each plot is marked with cross.

| Utility function | $\kappa$ | $\gamma$ | RMISE Mean ng/m$^3$ | RMISE SD ng/m$^3$ | Norm. RMISE Mean | Norm. RMISE SD |
|---|---|---|---|---|---|---|
| DUCB | 2.00 | 0.0 | 5.33 | 3.90 | 0.94 | 0.10 |
| auto-scaled DUCB | 1.50 | $-0.0001$ | 5.00 | 3.67 | 0.91 | 0.16 |
| PDUCB | 0.50 | $-10^{-9}$ | 4.19 | 2.99 | 0.75 | 0.24 |
| auto-scaled PDUCB | 1.00 | $-10^{-9}$ | 4.10 | 2.85 | 0.76 | 0.24 |
| GO | 0.10 | 0.0 | 5.40 | 3.88 | 0.95 | 0.09 |

| Utility function | $\kappa$ | $\gamma$ | WRMISE Mean ng/m$^3$ | WRMISE SD ng/m$^3$ | Norm. WRMISE Mean | Norm. WRMISE SD |
|---|---|---|---|---|---|---|
| DUCB | 2.00 | 0.0 | 3.48 | 3.04 | 0.91 | 0.22 |
| auto-scaled DUCB | 1.50 | $-0.0001$ | 3.28 | 2.95 | 0.88 | 0.26 |
| PDUCB | 0.50 | $-10^{-5}$ | 2.29 | 2.20 | 0.63 | 0.39 |
| auto-scaled PDUCB | 1.00 | $-10^{-7}$ | 2.37 | 2.33 | 0.64 | 0.37 |
| GO | 0.10 | 0.0 | 3.62 | 3.00 | 0.94 | 0.17 |

Table 6.2.: The minimal obtained error (RMISE and WRMISE) for each acquisition function and the parameter values used in the single source Gaussian dispersion scenario (D-NF-SS-SV).

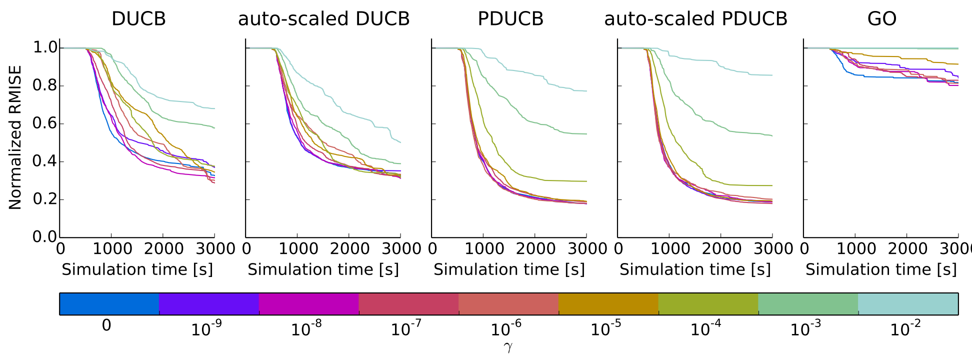


Figure 6.6.: The trial averaged normalized RMISE in the single source Gaussian scenario (G-NF-SS-SV) over simulation time. For each $\gamma$ the $\kappa$ resulting in the minimal final error was chosen. Each individual plot corresponds to one utility function and scaling. The auto-scaled versions use the scaling factor defined in Equations 3.30 and 3.31, in contrast to a constant scaling factor.

| Utility function | $\kappa$ | $\gamma$ | RMISE Mean ng/m$^3$ | RMISE SD ng/m$^3$ | Norm. RMISE Mean | Norm. RMISE SD |
|---|---|---|---|---|---|---|
| DUCB | 1.25 | 0.0 | 16.68 | 7.73 | 0.93 | 0.08 |
| auto-scaled DUCB | 2.00 | $-0.0001$ | 14.77 | 6.93 | 0.83 | 0.13 |
| PDUCB | 0.50 | 0.0 | 13.13 | 7.82 | 0.68 | 0.18 |
| auto-scaled PDUCB | 1.25 | $-10^{-6}$ | 13.20 | 7.87 | 0.68 | 0.19 |
| GO | 0.10 | 0.0 | 16.74 | 7.91 | 0.92 | 0.11 |

| Utility function | $\kappa$ | $\gamma$ | WRMISE Mean ng/m$^3$ | WRMISE SD ng/m$^3$ | Norm. WRMISE Mean | Norm. WRMISE SD |
|---|---|---|---|---|---|---|
| DUCB | 0.50 | $-10^{-5}$ | 11.35 | 6.58 | 0.93 | 0.14 |
| auto-scaled DUCB | 2.00 | $-0.0001$ | 9.66 | 6.11 | 0.80 | 0.21 |
| PDUCB | 1.00 | $-10^{-6}$ | 8.79 | 7.67 | 0.62 | 0.29 |
| auto-scaled PDUCB | 1.25 | $-10^{-5}$ | 8.44 | 7.02 | 0.65 | 0.26 |
| GO | 0.10 | 0.0 | 11.30 | 6.48 | 0.93 | 0.12 |

Table 6.3.: The minimal obtained error (RMISE and WRMISE) for each acquisition function and the parameter values used in the multiple source Gaussian dispersion scenario (D-NF-MS-SV).

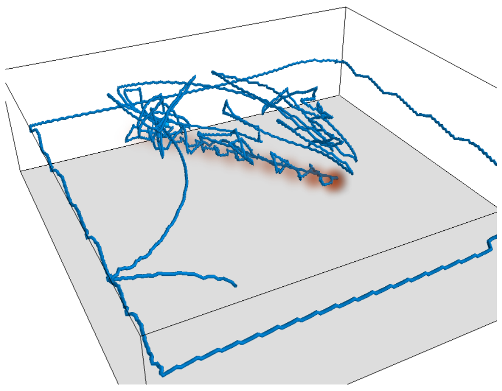


Figure 6.7.: Example of an UAV trajectory using the PDUCB acquisition function with $\kappa = 1.25$, $\gamma = -10^{-7}$, and automatic scaling in the single source Gaussian scenario. The plume is shown in red.

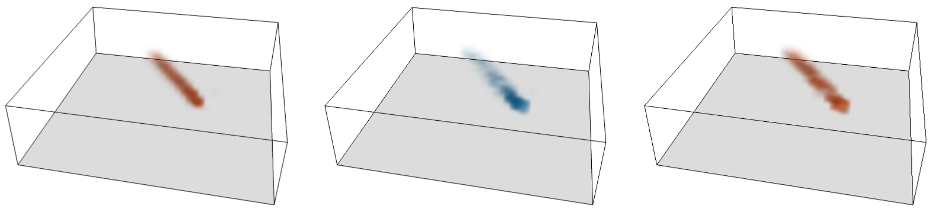


Figure 6.8.: One instance of the single source Gaussian dispersion scenario (D-NF-SS-SV). (a) Mean of the Gaussian process. (b) The absolute value of the difference of the predicted and true distribution. (c) The true plume distribution.

a too low $\kappa$ limits the exploration and let the UAV become stuck in a (local) maximum. The choice of $\gamma$ has no considerable effect as long as the distance penalty is not chosen too large ($\gamma < -10^{-5}$).

The same behavior for the choice of $\gamma$ is also observed for DUCB in the single source Gaussian scenario. However, this utility function is far more sensitive to the choice of $\kappa$. Using the scaling $s_{\mathrm{DUCB}}(\boldsymbol{y}) = 1$ the performance degrades setting $\kappa < 1$ and using the automatic scaling it degrades for $\kappa < 1.5$. In the dispersion scenarios the DUCB acquisition function does not perform well for any tested combination of parameter values.

Interestingly, DUCB performs slightly better with automatic scaling, whereas PDUCB performs slightly worse.

Finally, taking a look at the time course of error reduction (Figure 6.6) multiple phases can be discovered. About the first 500 s nearly no reduction occurs as in this phase the UAV only surround the area of interest. Then the error rapidly decreases in the next 500 s to 1000 s until the decrease levels off and stays almost constant for the rest of the simulation time.

These results show that PDUCB outperforms the DUCB and GO acquisition functions and in addition is robust against a non-optimal choice of parameters. The especially bad performance of the GO utility function is not too surprising as its intended use is to find a function maximum and not to build a correct model of the function (respectively plume concentration). DUCB works reasonable well for the simple case of a Gaussian distribution, but fails for the more localized dispersions.

The PDUCB performance might not seem to be too impressive in the dispersion scenarios, too. However, one has to keep in mind that even in Section 6.1 with much more samples the RMISE could not be reduced to less than 61 %. Also, qualitatively the plume is predicted at the correct location as Figure 6.8 shows, despite some deviance of the exact concentration values. The largest difference to the true distribution is close to the source at the concentration maximum.

Given this discussion I decided to limit further experiments to the PDUCB acquisition function. The other utility functions seem not to be a viable option (except maybe in the single source Gaussian scenario). As smaller values of $\kappa$ seem to provide slightly better results, but it should not be below 1 as argued, a value of $\kappa = 1.25$ was selected. The penalty distance seems to slightly improve the results up to $-10^{-5}$. However, to prevent to fall in the area where the performance then quickly decreases, it was set to a value of $\gamma = -10^{-7}$ which provided also good results. Both scaling approaches (constant or automatic) were used in further experiments as it is not clear from these results which one is better. Though, the automatic scaling is a bit worse in terms of error, it has one less parameter which has to be set according to the range of concentration values.

## 6.3. Evaluation in a Noisy Setting

It is important to verify that the methods for plume modeling also work in a noisy environment as in the real world noise will necessarily occur. For that a standard deviation of the sensor noise of $\sigma_{\text{sn}} = 10^{-5}\,\text{g/m}^3$ was assumed. As it should be possible to reliably estimate the noise level of the sensor, I considered this value as known. That allows to set the noise variance of the Gaussian process $\sigma_{\text{n}}^2 = \sigma_{\text{sn}}^2$. For a good prediction the ratio of $\sigma_{\text{n}}^2$ and the kernel variance $\sigma_{\text{k}}^2$ has to be chosen well. The process of doing that will be discussed in the next section before returning to the actual simulations.

### 6.3.1. Choosing the Kernel Variance

The method for determining the kernel process variance was essentially the same as described in Section 6.1 for determining the length scale. The points were it differs are that instead of the noiseless scenarios the single and multiple source plume dispersion scenarios with sensor noise (D-SN-SS-SV, D-SN-MS-SV) were used and only the Matérn kernel with $\nu = 3/2$ was used. Instead of varying the length scale it was fixed to $\ell = 5\,\text{m}$ and the kernel variance $\sigma_{\text{k}}^2$ was varied from $10^{-12}\,\text{g}^2/\text{m}^6$ to $10^{-3}\,\text{g}^2/\text{m}^6$.

The results are shown in Figure 6.9. The normalized error measures are highest (approaching 1) for low values of $\sigma_{\text{k}}^2$. For $\sigma_{\text{k}}^2 > 10^{-9}\,\text{g}^2/\text{m}^6$ the WRMISE stays the same, but the RMISE and QRSim reward slightly increase. This is less pronounced for multiple sources.

As all error measures do have their minimum at $\sigma_{\text{k}}^2 = 10^{-9}\,\text{g}^2/\text{m}^6$ or have almost reached it, this value has been used in the following.

### 6.3.2. Simulation of the Scenarios Including Sensor Noise

Using the kernel variance of $\sigma_{\text{k}}^2 = 10^{-9}\,\text{g}^2/\text{m}^6$ determined in the previous section the PDUCB method was evaluated in the single and multiple source plume dispersion scenario including sensor noise (D-SN-SS-SV, D-SN-MS-SV). The sen-

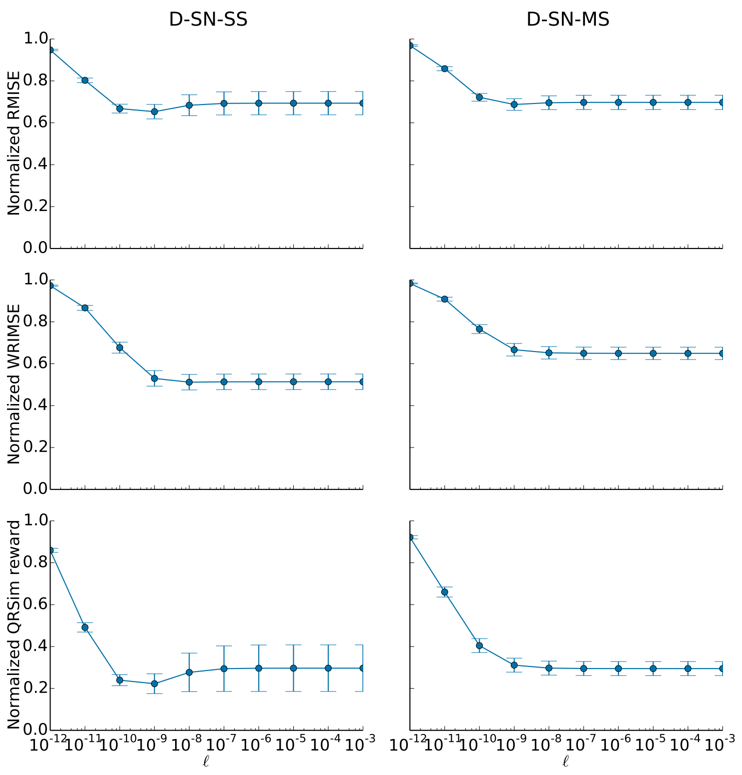


Figure 6.9.: The normalized error using the Matérn kernel ($\nu = 3/2$) in dependence of the kernel variance. The rows correspond to the RMISE, WRMISE, and QRSim reward error measures. The columns correspond to a single source Gaussian (G-SN-SS), and a multiple source Gaussian dispersion (D-SN-MS). Both scenarios were simulated with sensor noise. Error bars represent the standard error.

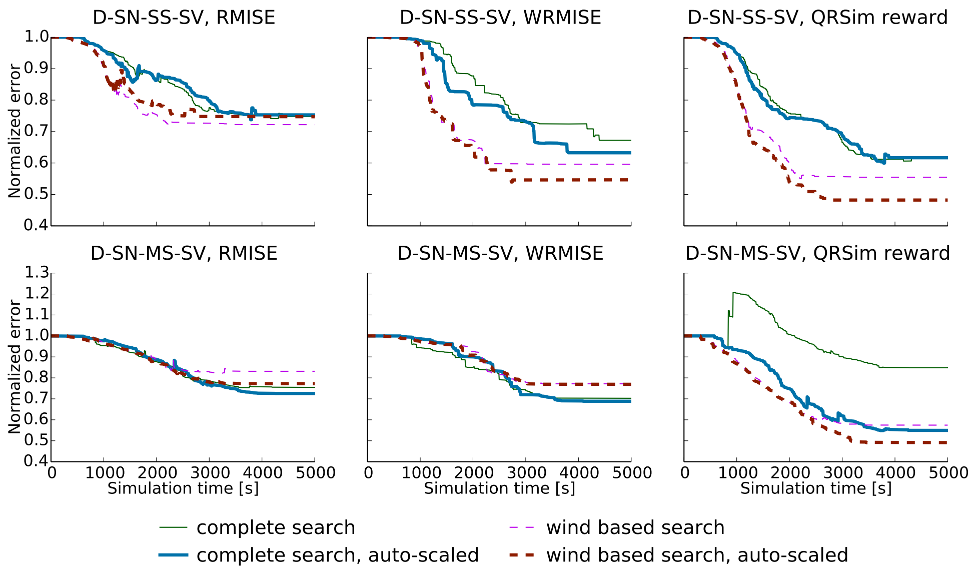


Figure 6.10.: Trial averaged normalized error over time in the single source dispersion (D-SN-SS-SV, top row) and multiple source dispersion scenario (D-SN-MS-SV, bottom row). Both scenarios include sensor noise. The columns correspond to the RMISE, WRMISE and the QRSim reward as error meausure. Solid lines show the results using the complete search method, whereas dashed lines used the wind based search. Thick lines show PDUCB with automatic scaling opposed to a constant scaling shown by the thin lines.

sor noise variance and the noise variance of the Gaussian process were set to $\sigma^2_{\mathrm{sn}} = \sigma^2_{\mathrm{n}} = 10^{-10}\,\mathrm{g}^2/\mathrm{m}^6$. As the change of $\sigma^2_{\mathrm{k}}$ (previously set to 1) influences $\sigma^2(\boldsymbol{x})$ the value of $\kappa$ has to be adjusted by the inverse factor. Hence, the PDUCB parameter setting in the following were $\kappa = 1.25 \cdot 10^9$ and $\gamma = -10^{-7}$.

In many instances surrounding the simulation area in just one height is not sufficient to discover the plume under the influence of noise. Thus, the complete search and the wind based search strategy described in Chapter 3.5.3 have been employed. The latter approach requires of course wind information which was read out from the QRSim simulator.

Apart from these points the same methods as in the noiseless case (Section 6.2) were used including the same number of 20 trials.

The results are shown in Figure 6.10 and Tables 6.4 and 6.5. All tested variants reduce the normalized RMISE to a value between 0.70 to 0.83 in both scenarios with a similar standard deviation around 0.17 to 0.24 in the single source scenario. Using the wind based search the error starts to decrease earlier in the single source scenario as less area has to be covered. Also the overall decrease seems to be a bit faster after the plume has bee discovered.

Looking at the WRMISE for a single source it turns out that the wind based

| | | RMISE | | Norm. RMISE | |
|---|---|---|---|---|---|
| Search method | Scaling | Mean ng/m$^3$ | SD ng/m$^3$ | Mean | SD |
| complete | constant | 4.06 | 2.60 | 0.75 | 0.24 |
| complete | auto | 4.16 | 2.83 | 0.75 | 0.23 |
| wind | constant | 3.90 | 2.43 | 0.72 | 0.21 |
| wind | auto | 3.95 | 2.69 | 0.75 | 0.23 |

| | | WRMISE | | Norm. WRMISE | |
|---|---|---|---|---|---|
| Search method | Scaling | Mean ng/m$^3$ | SD ng/m$^3$ | Mean | SD |
| complete | constant | 2.57 | 2.26 | 0.67 | 0.32 |
| complete | auto | 2.55 | 2.63 | 0.63 | 0.36 |
| wind | constant | 2.27 | 2.05 | 0.60 | 0.33 |
| wind | auto | 1.85 | 1.66 | 0.55 | 0.32 |

Table 6.4.: Trial averaged final error values in the noisy single source Gaussian scenario (D-SN-SS-SV).

| | | RMISE | | Norm. RMISE | |
|---|---|---|---|---|---|
| Search method | Scaling | Mean ng/m$^3$ | SD ng/m$^3$ | Mean | SD |
| complete | constant | 14.07 | 8.16 | 0.75 | 0.21 |
| complete | auto | 13.53 | 7.51 | 0.73 | 0.17 |
| wind | constant | 15.68 | 8.72 | 0.83 | 0.22 |
| wind | auto | 14.75 | 8.33 | 0.77 | 0.22 |

| | | WRMISE | | Norm. WRMISE | |
|---|---|---|---|---|---|
| Search method | Scaling | Mean ng/m$^3$ | SD ng/m$^3$ | Mean | SD |
| complete | constant | 9.02 | 7.36 | 0.70 | 0.29 |
| complete | auto | 13.53 | 7.51 | 0.73 | 0.17 |
| wind | constant | 9.99 | 6.96 | 0.77 | 0.24 |
| wind | auto | 14.75 | 8.33 | 0.77 | 0.22 |

Table 6.5.: Trial averaged final error values in the noisy multiple source Gaussian scenario (D-SN-MS-SV).

search decreases the normalized error by about 7 % additionally compared to the complete search. Thus, the wind based search is able to better approximate the actual plume without increasing the approximation error in other areas. With regard to the WRMISE the automatic scaling seems to perform a bit better (difference of 0.05 in the normalized error), but this is not the case for the RMISE. Given multiple sources the performance of the search strategies is almost equal. Also, the scaling method does not have a significant influence.

In the QRSim reward estimation occurs again an artifact in the multiple source scaling as already seen in Section 6.2 resulting in a normalized reward larger than one.

The results show that PDUCB can also be applied in a noisy setting. In comparison to a noiseless setting the error in the estimation slightly increases as would be expected given the same amount of training samples. However a more extensive search strategy has to applied in the beginning. Using wind information a good estimation might be obtained more quickly, but the effect is rather small. Both, a constant scaling and the automatic scaling perform equally. This makes the automatic scaling a better choice as it requires to set less parameters.

## 6.4. Multiple UAVs

Finally, the performance of the PDUCB acquisition function with the extension for multiples UAVs (Chapter 3.5.2) has been evaluated. For this exactly the same methods as in the previous section were used with exception of the scenario. This was replaced by the multiple UAV, multiple source plume dispersion scenario (D-SN-MS-MV). Also, a number of different $\rho \in \{10^{-i} | i = 5, \dots, 11\}$ has been tested.

Figure 6.11 shows the final normalized error in dependence of the value of $\rho$. Unfortunately, it is not possible to identify one value which is clearly better than others. Also, the normalized error is seldom decreased below 0.8. The best result is still obtained by the wind based search with auto-scaled PDUCB. This is most prominent for the WRMISE. As already in the first experiment the QRSim reward underestimates the error.

To understand why the performance of multiple UAVs is at best the same as of a single one it helps to directly compare the single and multiple UAV scenario as done in Figure 6.12. The decrease of the error stops earlier with multiple UAVs as the limit of training samples is faster reached with more vehicles sampling. As the error is decreasing only slightly faster with multiple UAVs the additional robots acquire samples in regions which do not add much to the overall estimation of the plume.

There are different reasons why that might be the case. The plume of a single source has a small spatial extent (except along the wind direction). Only one UAV can acquire samples in it without risking a collision with other UAVs. Moreover, the initial search strategy might be a problem. After one plume has been identified all vehicles switch to using the acquisition function. However, this makes the

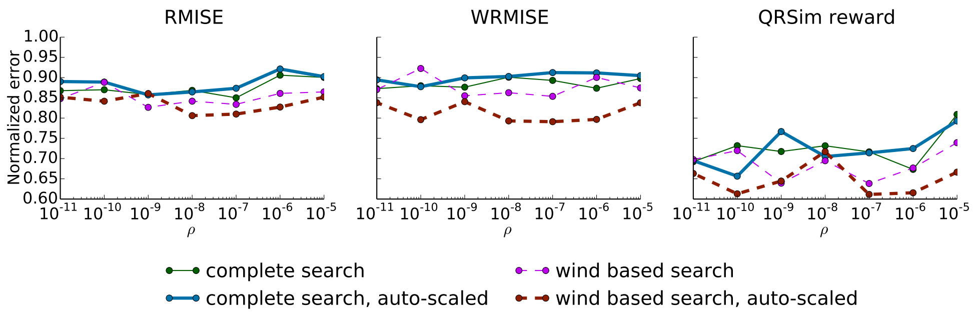


Figure 6.11.: The trial averaged final normalized error using multiple UAVs in the multiple source scenario (D-SN-MS-MV). Each plot corresponds to a different error measure. Solid lines show the results using the complete search method, where as dashed lines used the wind based search. Thick lines show PDUCB with automatic scaling opposed to a constant scaling shown by the thin lines.

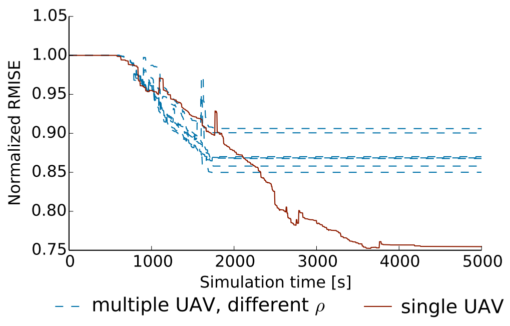


Figure 6.12.: Comparison of the average normalized RMISE over time for a single UAV (solid line) and multiple UAVs (dashed lines, each line is a different $\rho$). In both settings a total of 3000 samples was acquired.

discovery of further plumes somewhat random.

In conclusion there is definitely room for improvement regarding the usage of multiple UAVs.

# 7. Outlook on Time-varying Plumes

So far only distributions which are static over time have been discussed. This assumption might be violated in many instances. Unfortunately, time did not allow me to extend the proposed methods to time-varying plumes. Nevertheless, I provide some thoughts on how to do this.

The first thing to do is probably adding another input dimension to the Gaussian process representing time. This also requires adjusting the covariance function. As the temporal correlations might differ from the spatial correlations a product of a temporal and a spatial kernels would be a good start. Though that separability neglects potential spatio-temporal interdependencies, construction and hyper-parameter estimation is easier. Given wind with a strong directionality, a kernel modeling the spatio-temporal interdependencies becomes more important. Some work on separable as well as non-separable spatio-temporal covariance functions with application to environmental monitoring has been published by Singh et al. (2010).

Diffusion and advection by wind are the two main factors leading to a change of the plume distribution over time. While diffusion is a rather slow process, advection can occur on shorter time scales. Thus, it might be especially beneficial to include wind effects in the covariance function. Reggente and Lilienthal (2009) and Lilienthal et al. (2009) did this in another kernel based approach called Kernel DM+V/W algorithm.

A specific scenario with a time-varying plume distribution is suggested by De Nardi (2013). Instead of having a constant plume dispersion like in the scenarios discussed, each source emits puffs in random intervals travelling with the wind. Given a regular emission interval this could be modeled by using the value of a periodic function as time input or incorporating a periodic function into the covariance function. However, given a random emission interval this probably does not work as the frequency of the periodic function would have to change.

Another problem to be solved is locating such a puff dispersion. Already for the static dispersion an extensive search is required in the beginning. Given a puff dispersion measuring a low concentration could mean that the location does not lie in the path of the puff dispersion, but also that the measurement was taken between to puffs. Unfortunately, this seems like an inherent problem which cannot be completely solved.

Finally, let me discuss two properties of Gaussian processes to take into account when modeling time-varying plume distributions. When using a zero mean prior (as it is usually done), the predicted concentration mean will decay to zero over time (assuming $k(t, t') \to 0$ for $\left|t - t'\right| \to \infty$). However, in certain scenarios (like puff dispersions) it is likely that an increased concentration is measured again at a location in the future if this was the case once before at that location. It might be a good idea to adjust the mean prior based on the measurements to prevent a decay of the mean prediction. Note that the covariance function remains unchanged and the uncertainty at that location will still increase with time indicating that the measurement should be repeated.

The performance in time-varying scenarios might be impaired by the property that Gaussian processes have no notion of directionality of time. This follows from the (required) symmetry of the covariance function which does not allow to differentiate between $t < t'$ and $t > t'$. Hence, a kernel might model the path along which a plume travels, but not the direction. Along that direction the plume is likely to broaden because of dispersion. Respectively it gets more concentrated in the opposite direction. Effects like this cannot be modeled except for the overall statistics.

# 8. Conclusion

Gaussian processes have been used before in modeling of spatial data and environmental monitoring. However, previous approaches, namely DUCB, do not work well for plume dispersions as the simulation experiments have shown. Presumably, the steep concentration gradients and high locality are the main problem.

In this work DUCB has been adapted to PDUCB which could be shown to work reasonable well for plume dispersions. Nevertheless, there is still room for improvement as the error is seldom reduced by more than 30 % on average with a large variance. Also, an automatic scaling was introduced which allows choosing the PDUCB parameters independent of the actual concentrations. Finally, the PDUCB algorithm was extended for the use with multiple UAVs. However, this did not result in a better performance.

Besides the modeling of the plume distribution, localizing the dispersion at all is also an important problem. Noisy data does not allow to reliably estimate a concentration gradient. This requires the use of a systematic search approach. Incorporating information of the wind direction can speed up the search. Once a plume has been found PDUCB is able to map it quickly.

One problem that could not be solved in the scope of this thesis is an automatic selection of hyper-parameters based on the data. The usual approach of likelihood optimization clearly fails. Selecting hyper-parameters on a trial to trial basis would also allow a closer match of the prediction to the plume dispersion.

Besides that, the prediction quality might be improved if wind information is considered and included in the covariance function. Some pointers in that direction, though for a different algorithm, are given by Reggente and Lilienthal (2009). In general non-stationary kernels could improve the prediction, but they come at the cost of more hyper-parameters and prior assumptions about the plume dispersion which might be violated.

In summary, the basic applicability of Gaussian processe with a PDUCB acquisition function for plume distribution modeling has been shown. However, there is a number of possibilities of improvement left for future work. Also, it should be shown in future work that the proposed methods work in a real world scenario as only simulations have been performed. Further interesting research directions following from this work would be the inclusion and handling of obstacles or the modeling of time-varying plume distributions.

# A. Error Bound of a Mean Estimate

**Theorem 1.** *Let $y = y^* + \eta$ with $\eta \sim \mathcal{N}(0, \sigma^2)$ and $\bar{y}$ be the mean of $n$ samples from $y$. Then $n \geq 1.96^2 \cdot \left(\sigma/\rho\right)^2$ samples are needed to have the error bound $|\bar{y} - y^*| < \rho$ hold with probability $p \geq 0.95$.*

*Proof.* Given the error bound the true value $y^*$ has to, with probability $p$, lie in $[\bar{y} - \rho, \bar{y} + \rho]$. Thus, the 95 % confidence interval of $\bar{y}$ has to be a subset of that. With the standard error $\sigma/\sqrt{n}$ the confidence interval is obtained as

$$y^* \in \left[\bar{y} - 1.96 \cdot \frac{\sigma}{\sqrt{n}}, \bar{y} + 1.96 \cdot \frac{\sigma}{\sqrt{n}}\right] \subseteq \left[\bar{y} - \rho, \bar{y} + \rho\right].$$

From that follows

$$\rho \geq 1.96 \cdot \frac{\sigma}{\sqrt{n}} \quad \Leftrightarrow \quad n \geq 1.96^2 \left(\frac{\sigma}{\rho}\right)^2$$

$\square$

Assuming a Gaussian plume dispersion (Equation 2.2) with the highest concentration possible in the scenarios $Q = 2.5\,\mathrm{g/s}$, and $u = 3\,\mathrm{m/s}$, $\boldsymbol{s}' = (0, 0, -40\,\mathrm{m})^\top$ a concentration of $c(\boldsymbol{x}') \approx 0.055\,\mathrm{g/m^3}$ is obtained at $\boldsymbol{x}' = (10\,\mathrm{m}, 0, -40\,\mathrm{m})^\top$, a position in the center of the plume ten meters away from the source. For a usable measurement the magnitude of noise should be at least a magnitude lower, thus $\rho \leq 0.0055\,\mathrm{g/m^3}$. From the theorem it follows that, given the QRSim default noise standard deviation of $\sigma_{\mathrm{sn}} = 10^{-2}\,\mathrm{g/m^3}$ at least 385 samples are needed. Further away from the source or with a lower emission rate (which can also be a magnitude lower) even more samples would be needed.

# B. Sparse Online Gaussian Processes

In the following it will be proven that the matrix $-\boldsymbol{C}_t$ of a sparse online Gaussian process (Csató and Opper 2002) is symmetric, positive-definite. The proof allows to relate the rule for full updates to the update of the inverse Cholesky factor in Chapter 3.1. As the notation by Csató and Opper (2002) differs it should be noted that $[\sigma_x^2] = \boldsymbol{B}$ and $\boldsymbol{k}_{t+1} = K(X, \boldsymbol{x}_{t+1})$.

**Theorem 2.** *The matrix $-\boldsymbol{C}_t$ is symmetric, positive-definite for all $t \geq 1$.*

*Proof.* The proof is done by induction. It has to be shown

- that $-\boldsymbol{C}_1$ (base case) fulfills the proposition
- and that $-\boldsymbol{C}_{t+1}$ fulfills the proposition given it is fulfilled for $\boldsymbol{C}_t$ (inductive step).

The deletion of a basis vector has not to be considered as it exactly undoes a full update and then performs a reduced update.

**Base Case** For $t = 1$ we obtain

$$-\boldsymbol{C}_1 = \left[r^{(t+1)}\right] = \left[\sigma_x^{-2}\right]$$

As $\sigma_x > 0$ it follows that $-\boldsymbol{C}_1$ is symmetric positive-definite.

**Inductive Step** For showing the symmetry and positive-definiteness after a full update it suffices to show that Cholesky factorization for the updated matrix $-\boldsymbol{C}_{t+1} = \left(\boldsymbol{L}'^{-1}\right)^\top \boldsymbol{L}'^{-1}$ exists.

$$\begin{aligned}
-\boldsymbol{C}_{t+1} &= -U_{t+1}(\boldsymbol{C}_t) - r^{(t+1)} \boldsymbol{s}_{t+1} \boldsymbol{s}_{t+1}^\top \\
&= \begin{bmatrix} -C_t + \sigma_x^{-2} \boldsymbol{C}_t \boldsymbol{k}_{t+1} \boldsymbol{k}_{t+1}^\top \boldsymbol{C}_t^\top & \sigma_x^{-2} \boldsymbol{C}_t \boldsymbol{k}_{t+1} \mathbf{e}_{t+1}^\top \\ \sigma_x^{-2} \mathbf{e}_{t+1} k_{t+1}^\top \boldsymbol{C}_t^\top & \sigma_x^{-2} \end{bmatrix} \\
&= \begin{bmatrix} \left(\boldsymbol{L}^{-1}\right)^\top & \sigma_x^{-1} \boldsymbol{C}_t \boldsymbol{k}_{t+1} \\ 0 & \sigma_x^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}^{-1} & 0 \\ \sigma_x^{-1} \boldsymbol{k}_{t+1}^\top \boldsymbol{C}_t^\top & \sigma_x^{-1} \end{bmatrix} \\
&= \left(\boldsymbol{L}'^{-1}\right)^\top \boldsymbol{L}'^{-1}
\end{aligned}$$

In case of a reduced update the relation

$$-\boldsymbol{C}_{t+1} = -\boldsymbol{C}_t - r^{(t+1)} \boldsymbol{s}_{t+1} \boldsymbol{s}_{t+1}^\top$$

holds. The term $-r^{(t+1)}\boldsymbol{s}_{t+1}\boldsymbol{s}_{t+1}^\top$ is symmetric, positive-definite as it is an outer vector product (which is symmetric, positive-definite) multiplied by a positive number $-r^{(t+1)} = \sigma_x^{-2} > 0$. The sum $-\boldsymbol{C}_{t+1}$ of symmetric, positive-definite matrices is again symmetric, positive-definite. □

**Corollary 1.** *A full update of a symmetric, positive-definite matrix $-\boldsymbol{C}_t$ as formulated by Csató and Opper (2002, equation 2.9) consists of the same calculations as an online update of the inverse Cholesky factor in Equation 3.11.*

This is obvious from the inductive step for a full update.

# C. PDUCB Differentiability

**Theorem 3.** *The PDUCB acquisition function given by*

$$\begin{aligned}u_{\mathrm{PDUCB}}(\boldsymbol{x}) &= u_1(\boldsymbol{x}) + s_{\mathrm{PDUCB}}(\boldsymbol{y})u_2(\boldsymbol{x})\\ u_1(\boldsymbol{x}) &= (1-a)\cdot \ln\left(\mu_+(\boldsymbol{x})+\varepsilon\right) + a\cdot\ln\varepsilon\\ u_2(\boldsymbol{x}) &= \kappa\cdot\left(\sigma^2(\boldsymbol{x})-\sigma_{\mathrm{n}}^2\right)+\gamma\cdot d^2(\boldsymbol{x},\boldsymbol{x}')\end{aligned}$$

*with*

$$\begin{aligned}a &= \mathrm{e}^{-\mu_+(\boldsymbol{x})/\varepsilon}\\ \mu_+(\boldsymbol{x}) &= \max\left\{0,\mu(\boldsymbol{x})\right\}\end{aligned}$$

*is differentiable by $\boldsymbol{x}$ for all $\boldsymbol{x}$ if the Gaussian process providing $\mu(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x})$ is mean square differentiable.*

*Proof.* As

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}u_{\mathrm{PDUCB}}(\boldsymbol{x}) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}u_1(\boldsymbol{x}) + s_{\mathrm{PDUCB}}(\boldsymbol{y})\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}u_2(\boldsymbol{x})$$

it suffices to show the differentiability for $u_1(\boldsymbol{x})$ and $u_2(\boldsymbol{x})$ independently.

For $u_1(\boldsymbol{x})$ the derivative is

$$\begin{aligned}\tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}u_1(\boldsymbol{x}) &= \tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}(1-a)\cdot\ln\bigl(\mu_+(\boldsymbol{x})+\varepsilon\bigr)\\ &\quad + (1-a)\,\tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\bigl(\mu_+(\boldsymbol{x})\bigr)\,\frac{1}{\mu_+(\boldsymbol{x})+\varepsilon}\\ &\quad + \ln\varepsilon\cdot\tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}a\\ &= \tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\bigl(\mu_+(\boldsymbol{x})\bigr)\,a\varepsilon^{-1}\cdot\ln\bigl(\mu_+(\boldsymbol{x})+\varepsilon\bigr)\\ &\quad + (1-a)\,\tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\bigl(\mu_+(\boldsymbol{x})\bigr)\,\frac{1}{\mu_+(\boldsymbol{x})+\varepsilon}\\ &\quad - \ln\varepsilon\cdot\tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\bigl(\mu_+(\boldsymbol{x})\bigr)\,a\varepsilon^{-1}\\ &= \tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\bigl(\mu_+(\boldsymbol{x})\bigr)\cdot\left(a\varepsilon^{-1}\ln\bigl(\mu_+(\boldsymbol{x})+\varepsilon\bigr) - a\varepsilon^{-1}\ln\varepsilon + \frac{1-a}{\mu_+(\boldsymbol{x})+\varepsilon}\right)\\ &= \begin{cases}\tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\bigl(\mu(\boldsymbol{x})\bigr)\cdot\left(a\varepsilon^{-1}\ln\bigl(\mu(\boldsymbol{x})+\varepsilon\bigr)-a\varepsilon^{-1}\ln\varepsilon+\frac{1-a}{\mu(\boldsymbol{x})+\varepsilon}\right) & \mu(\boldsymbol{x})>0\\ \tfrac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}(0)\cdot\left(\varepsilon^{-1}\ln\varepsilon-\varepsilon^{-1}\ln\varepsilon+\frac{1-1}{\varepsilon}\right)=0 & \mu(\boldsymbol{x})\le 0\end{cases}.\end{aligned}$$

Hence, $u_1(\boldsymbol{x})$ is always differentiable for all $\boldsymbol{x}$ with $\mu(\boldsymbol{x})\le 0$. Furthermore, the parenthesized term is a composition of continuous functions for $\mu(\boldsymbol{x})>0$ and the derivative $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\mu(\boldsymbol{x})$ exists for a mean square differentiable Gaussian process. From that, the differentiability for all $\boldsymbol{x}$ follows.

For $u_2(\boldsymbol{x})$ the somewhat simpler derivative is

$$u_2(\boldsymbol{x}) = \kappa \cdot \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\sigma^2(\boldsymbol{x}) + \gamma \cdot \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\Big(d^2(\boldsymbol{x}, \boldsymbol{x}')\Big)$$

in which $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}\sigma^2(\boldsymbol{x})$ is differentiable for a mean square differentiable Gaussian process and the distance derivative is given by

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}d^2(\boldsymbol{x}, \boldsymbol{x}') = 2(\boldsymbol{x} - \boldsymbol{x}').$$

□

# D. Prior Width

**Theorem 4.** *To increase the log likelihood* $\ln p(\boldsymbol{y}|X,\boldsymbol{\theta},\mathcal{H}_j)$ *by at least* $\ln(\Delta p)$ *at* $\theta_i = m_{\theta_i}$ *with a Gaussian prior at* $m_{\theta_i}$ *the width* $\sigma_{\theta_i}$ *of this prior has to be lower or equal than* $1/(\Delta p\sqrt{2\pi})$*.*

*Proof.* The log likelihood combined with (independent) priors is given by

$$\ln\bigl(p(\boldsymbol{y}|X,\boldsymbol{\theta},\mathcal{H}_j)p(\boldsymbol{\theta}|\mathcal{H}_j)\bigr) = \ln p(\boldsymbol{y}|X,\boldsymbol{\theta},\mathcal{H}_j) + \sum_{i=1}^{n} \ln p(\theta_i|\mathcal{H}_j).$$

Hence, the condition

$$\ln p(\theta_i|\mathcal{H}_j) \geq \ln(\Delta p)$$

has to be fulfilled. Inserting the Gaussian prior it becomes:

$$\begin{aligned} N(m_{\theta_i}; m_{\theta_i}, \sigma_{\theta_i}) = \ln\left(\frac{1}{\sigma_{\theta_i}\sqrt{2\pi}}\right) &\geq \ln(\Delta p) \\ \Leftrightarrow \quad \frac{1}{\Delta p\sqrt{2\pi}} &\geq \sigma_{\theta_i} \end{aligned}$$

□

In Chapter 6.1 the difference of the maximum likelihood and the likelihood at $\ell = 5\,\mathrm{m}$ is $\ln(\Delta p) \approx 2078$ in the G-NF-SS scenario. To shift the likelihood maximum to $\ell = 5\,\mathrm{m}$ with a Gaussian prior with its mean at that position the width would need to be less than $\mathrm{e}^{-2078}/\sqrt{2\pi} \approx 0$.

# List of Figures

# List of Tables

# Bibliography

Björck, Å, H Park, and L Eldén (1994). "Accurate downdating of least squares solutions." In: *SIAM Journal on Matrix Analysis and Applications* 15.2, pp. 549–568.

Booker, A J et al. (1999). "A rigorous framework for optimization of expensive functions by surrogates." In: *Structural Optimization* 17.1, pp. 1–13.

Brochu, E, V M Cora, and N Freitas (2010). "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning." In: *arXiv.org*. arXiv: `1012.2599v1 [cs.LG]`.

Byrd, R H et al. (1995). "A Limited Memory Algorithm for Bound Constrained Optimization." In: *SIAM Journal on Scientific Computing* 16.5, pp. 1190–1208.

Chib, S and E Greenberg (1995). "Understanding the Metropolis-Hastings Algorithm." In: *The American Statistician* 49.4, pp. 327–335.

"Council Directive 96/62/EC of 27 September 1996 on ambient air quality assessment and management" (1996). In: *Official Journal of the EU* L 296, pp. 55–63.

Csató, L and M Opper (2002). "Sparse On-Line Gaussian Processes." In: *Neural Computation* 14.3, pp. 641–668.

De Nardi, R (2013). *The QRSim Quadrotor Simulator*. Tech. rep. RN/13/08. Department of Computer Science, University College London.

Gramacy, R B and H K H Lee (2008). "Bayesian Treed Gaussian Process Models With an Application to Computer Modeling." In: *Journal of the American Statistical Association* 103.483, pp. 1119–1130.

Guestrin, C, A Krause, and A Singh (2005). "Near-optimal Sensor Placements in Gaussian Processes." In: *22nd International Conference on Machine Learning*. Bonn, Germany: ACM, pp. 265–272.

Jatmiko, W, K Sekiyama, and T Fukuda (2007). "A PSO-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement." In: *Computational Intelligence Magazine, IEEE* 2.2, pp. 37–51.

Lilienthal, A J et al. (2009). "A statistical approach to gas distribution modelling with mobile robots – The Kernel DM+V algorithm." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. St. Louis, USA: IEEE.

Marchant, R and F Ramos (2012). "Bayesian optimisation for Intelligent Environmental Monitoring." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura, Algarve, Portugal: IEEE, pp. 2242–2249.

Neal, R M (1997). "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification." In: *arXiv.org*. arXiv: `arXiv:physics/9701026v2 [physics.data-an]`.

Oliphant, T E (2007). "Python for Scientific Computing." In: *Computing in Science & Engineering* 9.3, pp. 10–20.

Osborne, M A, R Garnett, and S J Roberts (2009). "Gaussian processes for global optimization." In: *3rd International Conference on Learning and Intelligent Optimization*.

Osborne, M A, S J Roberts, et al. (2008). "Towards Real-Time Information Processing of Sensor Network Data Using Computationally Efficient Multi-output Gaussian Processes." In: *International Conference on Information Processing in Sensor Networks*. IEEE, pp. 109–120.

Pedregosa, F et al. (2011). "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Petersen, K B and M S Pedersen (2008). *The Matrix Cookbook*.

Quiñonero-Candela, J and C E Rasmussen (2005). "A unifying view of sparse approximate Gaussian process regression." In: *The Journal of Machine Learning Research* 6, pp. 1939–1959.

Rasmussen, C E and C K I Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Reggente, M and A J Lilienthal (2009). "Using local wind information for gas distribution mapping in outdoor environments with a mobile robot." In: *IEEE Sensors*. Christchurch: IEEE, pp. 1715–1720.

Sacks, J et al. (1989). "Design and Analysis of Computer Experiments." In: *Statistical Science* 4.4, pp. 409–423.

Scott, D W (1992). *Multivariate Density Estimation*. Theory, Practice, and Visualization. New York, Chicester: John Wiley & Sons.

Settles, B (2009). *Active Learning Literature Survey*. Tech. rep. 1648. University of Wisconsin-Madison.

Singh, A et al. (2010). "Modeling and decision making in spatio-temporal processes for environmental surveillance." In: *IEEE International Conference on Robotics and Automation*. Anchorage, Alaska, USA: IEEE, pp. 5490–5497.

Stachniss, C et al. (2008). "Gas Distribution Modeling using Sparse Gaussian Process Mixture Models." In: *Robotics Science and Systems*.

Stockie, J M (2011). "The Mathematics of Atmospheric Dispersion Modeling." In: *SIAM review* 53.2, pp. 349–372.

Stranders, R, A Rogers, and J Jennings (2008). "A decentralized, on-line coordination mechanism for monitoring spatial phenomena with mobile sensors." In: *Second International Workshop on Agent Technology for Sensor Networks*, pp. 9–15.

Wang, H et al. (2010). "Plume Source Localizing in Different Distributions and Noise Types Based on WSN." In: *International Conference on Communications and Mobile Computing*. IEEE, pp. 63–66.

Zarzhitsky, D, D F Spears, and W M Spears (2005). "Distributed robotics approach to chemical plume tracing." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4034–4039.

Zhu, C et al. (1997). "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization." In: *ACM Transactions on Mathematical Software (TOMS)* 23.4, pp. 550–560.