Master's Thesis
MSC-044

Mechanik und Meerestechnik

Sampling-based Informative Path Planning for Field Exploration with micro Underwater Robots

by
Matti Vahs

Supervisors:
Prof. Dr.-Ing. Robert Seifried
Prof. Dr.-Ing. Thorsten A. Kern
Daniel-André Dücker, M.Sc.

Hamburg University of Technology
Institute of Mechanics and Ocean Engineering
Prof. Dr.-Ing. R. Seifried

Hamburg, May 2021

Contents

Chapter 1

Introduction

Autonomous exploration using mobile robots has been intensively investigated in the last decades [ThrunEtAl04, MalliosEtAl14]. In such exploration scenarios, an autonomous robot can be used for mapping or monitoring of environmental processes which can be summarized as gathering information about the robot's environment. Hereby, the robot's overall goal is to maximize some information metric, e.g. the accuracy of the inspection or the probability of locating some target in the environment. There is a variety of possible applications of autonomous exploration, including magnetic field mapping on ground, wind field mapping in the air, or leakage detection underwater. While many autonomous missions can be completed using unmanned aerial vehicles (UAVs) or unmanned ground vehicles (UGVs), the underwater domain is of particular interest, as using autonomous underwater vehicles (AUVs) makes it possible to investigate one of the least explored places on earth: The ocean. This is of enormous interest as, according to the National Oceanic and Atmospheric Administration (NOAA), more than 80 percent of our oceans are unmapped, unobserved, and unexplored. As an example, a fleet of AUVs equipped with sonars could be used to autonomously build topographic maps of the oceans seafloor.

On the other hand, small and low cost micro autonomous underwater vehicles (μAUVs) can be used to operate in small-scale, spatially constrained environments such as industrial tanks or underwater caves. Possible fields of applications for μAUVs are among others the monitoring of processes in industrial plants or inspection and mapping of groins which are built perpendicular to the shore and are used for coastal protection. Figure 1.1 provides an example of a μAUV named HippoCampus that was developed at the Institute of Mechanics and Ocean Engineering at Hamburg University of Technology.

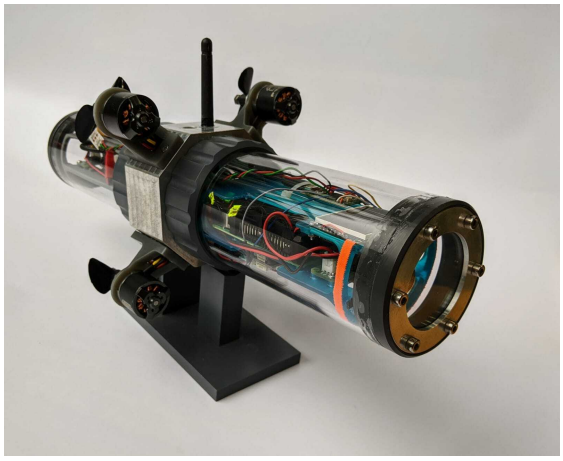


Figure 1.1: The HippoCampus μAUV platform.

1.1 Motivation

Consider the scenario, where a hazardous pollutant flows unhindered into the ocean as a consequence of a pipeline leakage below sea level. To prevent dramatic consequences for the environment, the leakage location should be detected as fast as possible. Since the pollutant and the potentially unknown environment can be a threat for human divers, the use of AUVs can be of particular advantage. A fleet of AUVs equipped with sensors can be used to build a map of the pollutant distribution based on taken measurements. Then, areas of high concentrations can be located which represent the source of the pollutant. To increase the time efficiency of the exploration, the robots plan their measurement locations in such a way that the information along a path is maximized with respect to some metric. Possible metrics could be, for example, the accuracy in areas of high concentration or the overall uncertainty reduction in the concentration map.

To summarize the goal of using AUVs for an underwater field exploration, an initially unknown environmental field can be estimated with an autonomously exploring robot that plans its path to achieve a mission specific goal. Therefore, an intelligent field exploration algorithm is needed which can be subdivided in two main modules. First, a belief model is required which represents the robot's internal knowledge about the state of the environment. This belief is updated sequentially as new measurements and their corresponding locations are available to generate a representation of the underlying environmental field. Second, an informative path based on the current field belief is planned which is suitable according to imposed constraints such as consumed energy and obstacle avoidance. The planned path which is tracked by a low level controller of the AUV should contribute to the overall mission goal. The interaction of the individual modules that are required for a field exploration is illustrated in Fig. 1.2

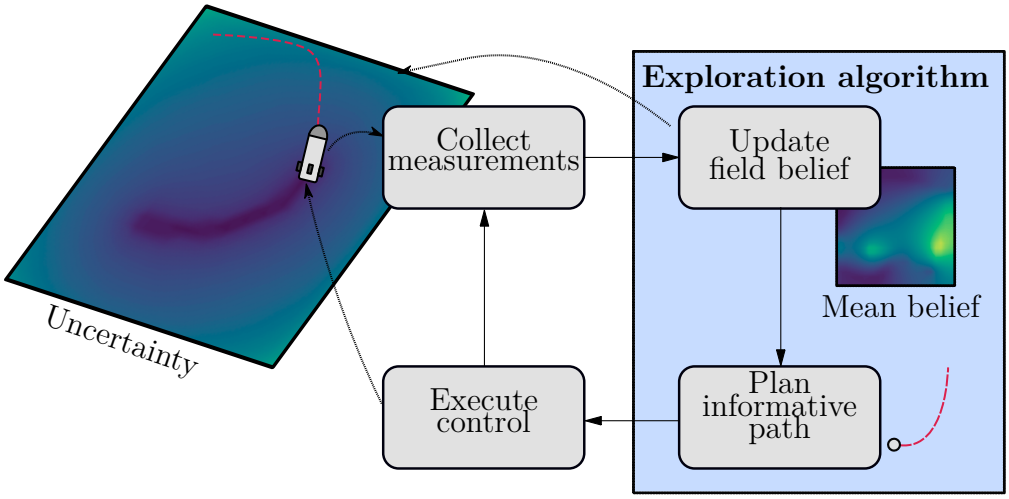


Figure 1.2: Interaction of the individual modules needed for an autonomous exploration of an environmental field. The exploration algorithm which is designed in this thesis is highlighted in light blue.

1.2 Problem Statement

In order to draw conclusions about an environmental field, an accurate map is needed which represents the underlying process sufficiently well. The true field that is to be estimated can be any scalar field in two dimensions such as an underwater temperature distribution at a specific water depth, a topographic map or a distribution of oil spills at the water surface. As motivated before, AUVs are highly suitable for such exploration and mapping tasks since they can also operate in hazardous areas. With the aim of creating an accurate map of the environmental field, the robot needs to collect measurements in large parts of the operating space. Creating a map using an autonomous robot can be achieved by application of an exploration algorithm consisting of a field belief (**Module I**) and an informative path planner (**Module II**). A design of these two modules which enable an autonomous field exploration is addressed in this thesis.

As underwater communication is often not reliable, a link to a central computer is not feasible and therefore all of the computations for modeling the field and planning the robot's path have to be carried out on-board. This imposes several restrictions on the exploration modules such as limited computational complexity and memory of the proposed algorithms.

In addition to the computational constraints, further requirements of the field belief representation have to be met. As the robot is collecting more and more measurements over time, the field belief has to be updated sequentially, since

newly arriving measurements can affect the planning behavior. Moreover, a measure of uncertainty should be provided in the belief model which allows for a more robust estimation of the environment's state. Therefore, probabilistic belief models should be taken into consideration as they meet the two aforementioned requirements.

The planning module which generates informative paths has to be adaptable to the user defined mission goals such as the motivating example of source localization. To achieve these goals, a global coverage of the area of interest is of utmost importance in order to minimize the overall uncertainty in the field belief. Therefore, longer planning horizons are needed since they enable the robot to early guide the exploration to areas of high uncertainty. Further, the planning module has to be able to generate paths sufficiently fast to replan the robot's path if necessary as in the case of suddenly occurring obstacles. Promising planning methods for an underwater field exploration are therefore the sampling-based path planners since they allow for long planning horizons as well as fast replanning. Sampling-based path planners can be designed in an anytime fashion meaning that a feasible path is quickly identified and improved with increasing computation time. In this way, the "quality" of the generated paths can always be scaled with computation time and thus, the path planner is applicable to computationally constrained platforms like the HippoCampus.

To sum up, this thesis aims to answer the fundamental question on how to determine a path of measurement locations that enables the exploration of environmental fields. This question is answered in two parts in this thesis. First, a mathematical representation of the environmental field is elaborated which is used to, second, plan an informative path for the exploration. Furthermore, the applicability of the proposed algorithms to computationally constrained platforms must always be given. Therefore, both modules of the exploration algorithm have to be designed such that they can be applied to μAUVs like the HippoCampus.

1.3 Contributions

In this thesis, an algorithm for the estimation of environmental fields is designed which is then utilized to enable informative path planning.

In **Module I**, Gaussian Markov Random Fields (GMRFs) are reviewed as potential field belief representation of two-dimensional fields. GMRFs are compared to other state of the art belief models in terms of criteria that are considered important for an underwater exploration.

In **Module II**, the second main part of this thesis, a novel approach to the informative path planning problem is presented. An extension of the

real-time Rapidly-exploring Random Tree (RT-RRT*) algorithm, presented in [NaderiRajamakiHamalainen15], is proposed in this work. To enable autonomous exploration, the RT-RRT* algorithm is extended by an informative framework which utilizes GMRFs in order to generate paths of potential measurement locations. The outcome is a highly optimized informative path planning algorithm that, in contrast to other state of the art informative path planners, quickly provides paths which are replanned while the robot is following a path. Further, fast replanning enables exploration of cluttered environments where a robot has to adapt its path whenever an obstacle suddenly occurs.

The performance of the proposed path planning algorithm in combination with GMRFs is analyzed in simulations. First, variations of the most influential tuning parameters are examined. Second, a case study is conducted which considers the different exploration scenarios of source localization and exploration in cluttered environments.

1.4 Outline

The remainder of this thesis is structured as follows. First, fundamental concepts in probability are reviewed in Chapter 2. In particular, multivariate Gaussian densities as well as Gaussian Processes are reviewed as prerequisite for the field modeling algorithm. The two modules of the autonomous exploration algorithm are separated in individual chapters since the field modeling is independent of the planning module. Chapter 3 presents the Gaussian Markov Random Fields (GMRFs) as approximation of a Gaussian Process. To justify the selection of GMRFs as field representation, it is compared to other state of the art belief models in terms of accuracy and computation time. The informative path planning module is covered in Chapter 4. A modification of the anytime motion planner RT-RRT* is used to solve the informative path planning problem. This algorithm is especially intended for real-time applications and is thus highly suitable for an environmental field exploration. The resulting path planner is analyzed in simulations in Chapter 5. Besides from the performance analysis, the exploration algorithm is applied to special cases which are particularly challenging. Finally, the work of this thesis is summarized in Chapter 6 and possible future directions are presented.

Chapter 2

Fundamentals

This chapter reviews fundamental concepts in probability which are used throughout this thesis. First, multivariate Gaussian distributions are discussed which are essential for many belief models such as Gaussian Processes or Kalman Filters. Especially Gaussian Processes are of great interest, since these are commonly used to describe unknown nonparametric functions over space or time given some (potentially noisy) observations of this function. Gaussian Processes can be applied to a variety of cases such as system identification, classification or environmental field modeling. Therefore, Gaussian Processes are explained in depth since they lay the foundation for the environmental field belief description which is presented in Chapter 3. Lastly, the concept of weighted shape functions is reviewed in order to improve the proposed field belief model.

2.1 Multivariate Gaussian Distributions

In this section, the concepts of multivariate Gaussian distributions are presented to the reader. Some basic knowledge about probability and stochastic processes is assumed to be known. However, for an in depth explanation of the covered concepts, the reader is referred to [Murphy13]. The core of the field belief model in Chapter 3 addresses the estimation of environmental field values in continuous space, meaning that the field value is characterized by a random variable X that can take values from a continuous space. A commonly used function to describe the density of such a random variable is the univariate Gaussian distribution which is characterized by its mean value $\mu = \mathbb{E}(X)$ and its variance $\sigma^2 = \mathrm{Var}(X)$. The univariate Gaussian distribution, often also called normal distribution, is

given by the function

$$p(x) = \mathcal{N}\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right), \tag{2.1}$$

where $(x;\cdot)$ describes the distribution of x given mean and covariance $(\cdot)$. Throughout this thesis the short hand notation $X \sim \mathcal{N}(\mu, \sigma^2)$ is often used to describe that the random variable X is Gaussian distributed with mean μ and variance σ^2. The Gaussian distribution in Eq. (2.1), however, assumes that x is a scalar value which is often not sufficient, as in the case of spatially distributed random variables of an environmental field. In this case, the probability density function (PDF) can be generalized for the multi-dimensional random variable $\boldsymbol{x} \in \mathbb{R}^n$, reading

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{2.2}$$

Here, $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean vector, $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ is a positive semidefinite and symmetric matrix called *covariance matrix* and $|\cdot|$ denotes the determinant. This parametrization of the PDF is also known as the moment parametrization with the first order moment being the mean vector and the second order moment being the covariance matrix. A second equivalent form of parametrization which is used subsequently is the *canonical* or $information\ form$

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\xi}, \boldsymbol{\Omega}) = \eta \exp\left(-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Omega} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{\xi}\right), \tag{2.3}$$

with the canonical parameters $\boldsymbol{\xi} = \boldsymbol{\Omega}\boldsymbol{\mu}$ and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$, and the normalizing constant η which ensures that

$$\int \mathcal{N}(\boldsymbol{x}; \boldsymbol{\xi}, \boldsymbol{\Omega})\, \mathrm{d}x = 1. \tag{2.4}$$

The canonical parameter $\boldsymbol{\Omega}$ is also called $information$ or *precision matrix*.

Having defined the multivariate Gaussian distribution, two important concepts for inference of random variables are presented hereafter. Here, the term inference refers to the process of drawing conclusions from the underlying probability distribution.

2.1.1 Marginalization

Whenever a subset of a collection of random variables is of interest, marginalization can be utilized to obtain a marginal distribution from a multivariate

distribution. Marginalization is used in this thesis to obtain the probability distribution of a field value at a specific location, given the multivariate Gaussian distribution of the entire field.

Let the PDF over the random variables $p(\boldsymbol{x}) = p\left(\left[\boldsymbol{x}_1, \boldsymbol{x}_2\right]^T\right)$ be a joint Gaussian distribution of the form $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \qquad \boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1} = \begin{bmatrix} \boldsymbol{\Omega}_{11} & \boldsymbol{\Omega}_{12} \\ \boldsymbol{\Omega}_{21} & \boldsymbol{\Omega}_{22} \end{bmatrix}^{-1} \tag{2.5}$$

are the joint mean vector and covariance matrix of the random vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Note that $\boldsymbol{\Omega}_{12} = \boldsymbol{\Omega}_{21}$ holds for the information of the two random vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ since $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}$ is symmetric. The marginal distribution of a random variable $p(\boldsymbol{x}_1)$ can be obtained by integrating the joint distribution over all possible outcomes of $\boldsymbol{x}_2$, reading

$$p(\boldsymbol{x}_1) = \int p(\boldsymbol{x}_1, \boldsymbol{x}_2)\, \mathrm{d}\boldsymbol{x}_2. \tag{2.6}$$

In the Gaussian case, this can be expressed in closed form yielding the marginal densities

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \tag{2.7}$$
$$p(\boldsymbol{x}_2) = \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}), \tag{2.8}$$

which are again Gaussian distributed. A detailed derivation of the marginal densities of multivariate Gaussian distributions can be found in [Murphy13]. Figure 2.1 depicts an exemplary two-dimensional Gaussian distribution with its marginal distributions illustrated at the borders. It is important to see that the marginal density $p(\boldsymbol{x}_1)$ is independent of $\boldsymbol{x}_2$ and vice versa.

2.1.2 Conditioning

Another major concept for inference is the conditioning of random variables. Oftentimes, random variables carry some information about other random variables and their knowledge can be used to improve the marginal PDFs. Incorporating this knowledge yields the *conditional* or *posterior* probability distribution.

The conditional probability of a random variable $\boldsymbol{x}_1$ given $\boldsymbol{x}_2$ is defined as their joint probability distribution divided by the marginal distribution of $\boldsymbol{x}_2$, hence

$$p(\boldsymbol{x}_1 | \boldsymbol{x}_2) = \frac{p(\boldsymbol{x}_1, \boldsymbol{x}_2)}{p(\boldsymbol{x}_2)}. \tag{2.9}$$

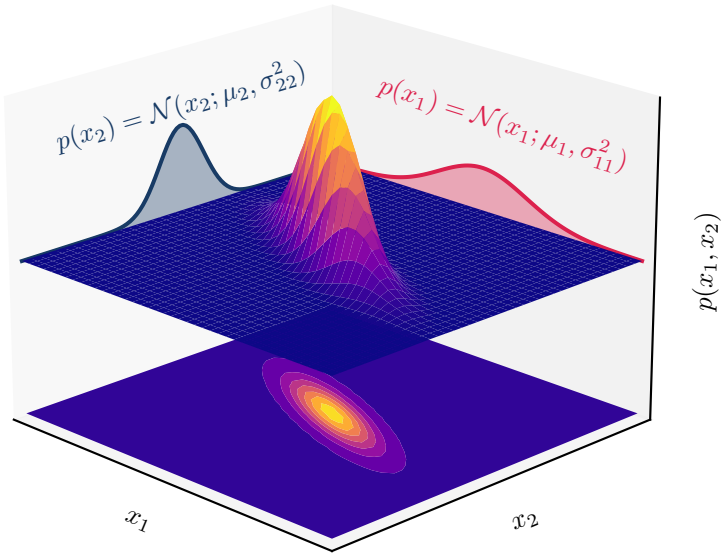


Figure 2.1: An exemplary two-dimensional Gaussian joint distribution $p(x_1, x_2)$ with the marginal distributions $p(x_1)$ and $p(x_2)$.

This definition of conditional probability was introduced by the mathematician Andrei Kolmogorov in 1931 and is still one of the most important tools in probability. In the Gaussian case, the posterior distribution in Eq. (2.9) is in fact again a Gaussian PDF of the form

$$p\left(\boldsymbol{x}_1|\boldsymbol{x}_2\right) = \mathcal{N}\left(\boldsymbol{x}_1; \boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}\right) \tag{2.10}$$

with the conditional mean $\boldsymbol{\mu}_{1|2}$ and covariance matrix $\boldsymbol{\Sigma}_{1|2}$. Using the previously introduced marginalization of Gaussian densities, the conditional moments can be obtained by substitution of Eq. (2.2) and Eq. (2.8) in Eq. (2.9) and application of some basic linear algebra operations, yielding

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\left(\boldsymbol{x}_2 - \boldsymbol{\mu}_2\right) \tag{2.11}$$

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}. \tag{2.12}$$

As one may notice, conditioning in particular can be computationally expensive since it depends on the inverse of the marginal covariance $\boldsymbol{\Sigma}_{22}$ which can be of arbitrarily high dimension. Therefore, it is often preferred to use the canonical

parametrization in Eq. (2.3) in order to obtain the conditional distribution

$$p\left(\boldsymbol{x}_1|\boldsymbol{x}_2\right) = \mathcal{N}\left(\boldsymbol{x}_1; \boldsymbol{\xi}_{1|2}, \boldsymbol{\Omega}_{1|2}\right) \tag{2.13}$$

with the canonical parameters

$$\boldsymbol{\xi}_{1|2} = \boldsymbol{\xi}_1 - \boldsymbol{\Omega}_{12}\boldsymbol{x}_2 \tag{2.14}$$

$$\boldsymbol{\Omega}_{1|2} = \boldsymbol{\Omega}_{11} \tag{2.15}$$

which do not explicitly depend on an inverse matrix. However, the main disadvantage of the canonical representation is that the marginalization of a random variable relies on the inversion of the information $\boldsymbol{\Omega}_{22}$. Note, that the opposite holds for the moment parametrization.

A geometric and intuitive interpretation of conditioning can be found in Fig. 2.2. Essentially, the conditioning of a random variable is nothing more than cutting a plane through the multivariate PDF at a specific value and looking at it from the side. Fig. 2.2 shows the conditional distributions of x_1 given x_2 and vice versa. The typical Gaussian shape is clearly identifiable in the conditional distribution.

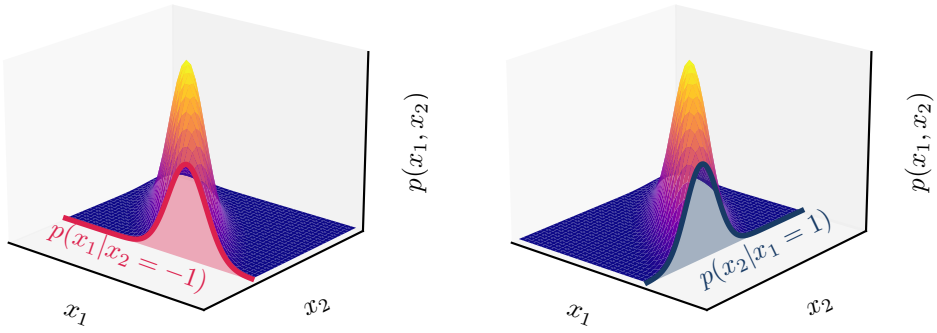


Figure 2.2: Illustration of conditioning in the two-dimensional Gaussian case. The conditional distributions, highlighted in the plots, are given by the intersection of the joint distribution $p\left(x_1, x_2\right)$ and the planes $\{\boldsymbol{x} \in \mathbb{R}^2 : x_1 = 1\}$ and $\{\boldsymbol{x} \in \mathbb{R}^2 : x_2 = -1\}$, respectively.

2.2 Bayesian Inference

Bayesian inference is a method in statistics to update the probability distribution of a random variable as more information becomes available. Its name

originates from Bayes' theorem or Bayes' rule which relates the posterior distribution $p(\boldsymbol{x}_1|\boldsymbol{x}_2)$ to its "inverted" PDF $p(\boldsymbol{x}_2|\boldsymbol{x}_1)$, see [ThrunBurgardFox05]. In the continuous case Bayes' rule is given by

$$p(\boldsymbol{x}_1|\boldsymbol{x}_2) = \frac{p(\boldsymbol{x}_2|\boldsymbol{x}_1)\, p(\boldsymbol{x}_1)}{p(\boldsymbol{x}_2)} \tag{2.16}$$

which can be further simplified. The denominator is a constant since $\boldsymbol{x}_2$ is given and hence the marginal PDF $p(\boldsymbol{x}_2)$ can be evaluated. This is often also written as

$$p(\boldsymbol{x}_1|\boldsymbol{x}_2) = \eta\, p(\boldsymbol{x}_2|\boldsymbol{x}_1)\, p(\boldsymbol{x}_1) \tag{2.17}$$

$$\Leftrightarrow p(\boldsymbol{x}_1|\boldsymbol{x}_2) \propto p(\boldsymbol{x}_2|\boldsymbol{x}_1)\, p(\boldsymbol{x}_1) \tag{2.18}$$

where η is a normalizing constant and $\propto$ stands for "proportional to". In many robotic applications $\boldsymbol{x}_1$ is considered to be the state of the system that one wants to estimate and $\boldsymbol{x}_2$ are the observations that are made by sensors. Then, $p(\boldsymbol{x}_1|\boldsymbol{x}_2)$ is the posterior distribution which includes the most recent observation, $p(\boldsymbol{x}_2|\boldsymbol{x}_1)$ is the measurement probability and $p(\boldsymbol{x}_1)$ is the *prior* distribution. In the Gaussian case, the posterior distribution is roughly speaking a weighted sum of the measurement probability and the prior based on their variance. In this work, Bayesian inference is used to update the probability distribution of field values based on taken measurements.

2.3 Gaussian Processes

The Gaussian Process (GP) is a stochastic process which is used as a regression method and finds vast application in the field of machine learning. Hereby, the main goal is to provide accurate predictions of an unknown function at unobserved points given some observed data. GPs are based on the concepts of multivariate Gaussian distributions. However, rather than having a distribution of a random variable, GPs define a distribution over functions [Kocijan16]. In the following, the mathemetical properties of GPs are introduced in order to better understand the concepts of the field belief description used in this thesis.

As previously mentioned, GPs are used to approximate an unknown function $f(\boldsymbol{x})$ over space such that one can predict the function value f_i at the input location $\boldsymbol{x}_i$. This can be done by modeling the values f_i as random variables and stacking them in a vector $\boldsymbol{f}$ which forms a joint Gaussian distribution. The associated PDF is given by

$$p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}; \boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x})) \tag{2.19}$$

where $\boldsymbol{\mu}(\boldsymbol{x})$ is the mean vector and $\boldsymbol{\Sigma}(\boldsymbol{x})$ is the covariance matrix which can be constructed using a *covariance* or *kernel function* $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This function is used to model the spatial dependency of two input locations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The corresponding entry of the covariance matrix is given by

$$\boldsymbol{\Sigma}_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.20}$$

According to [RasmussenWilliams06], a GP is fully specified by its mean function

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})] \tag{2.21}$$

and its covariance function

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}[(\mu(\boldsymbol{x}_i) - f(\boldsymbol{x}))(\mu(\boldsymbol{x}_j) - f(\boldsymbol{x}))]. \tag{2.22}$$

All together, the GP can be written as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)). \tag{2.23}$$

Inspecting the given equations, it becomes apparent that the performance of a GP heavily depends on the choice of covariance function which is why two common kernel functions are presented in the following.

2.3.1 Kernel Functions

The kernel or covariance function $\kappa : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ defines the covariance of two random variables $cov(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j))$ at the input locations $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ where $\mathcal{X}$ is the set of input locations. As one can see, the kernel function relates the covariance of the outputs of a GP to their inputs. The basic idea behind covariance functions is that input locations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that are close in input space are likely to also have close values in output space. An important requirement for a valid kernel function is

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i) \geq 0, \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X} \tag{2.24}$$

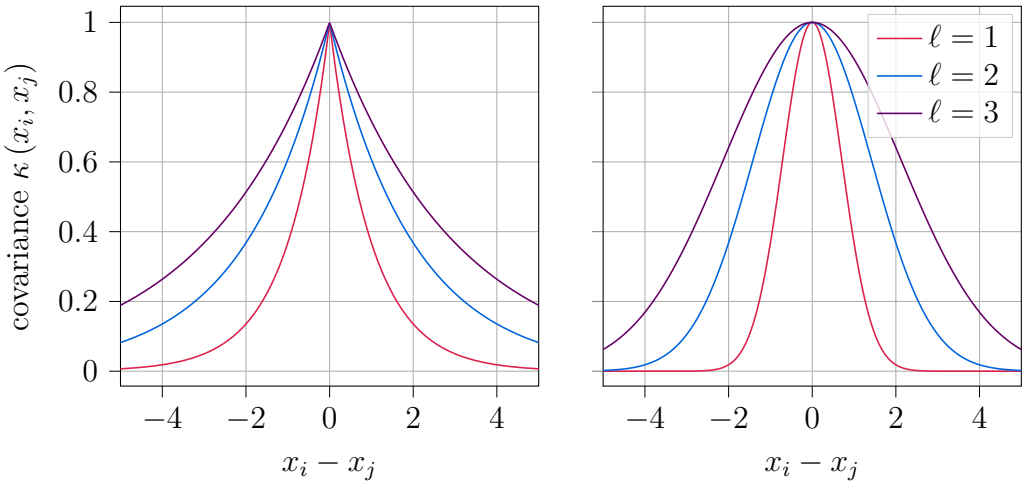
which states that the kernel is symmetric and positive semidefinite. This property ensures that the resulting covariance matrix is also symmetric and positive semidefinite. As stated in [RasmussenWilliams06], a kernel is said to be *stationary* if the covariance of two input locations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ only depends on their difference $(\boldsymbol{x}_i - \boldsymbol{x}_j)$. Furthermore, a kernel is said to be *isotropic* if the covariance only depends on the Euclidean distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. Subsequently, two isotropic kernel functions are presented which are frequently used in the literature.

The γ-Exponential Kernel

The family of γ-exponential kernel functions is given by

$$\kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sigma_f^2 \exp\left(-\left(\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\ell}\right)^{\gamma}\right), \quad 0 < \gamma \leq 2 \tag{2.25}$$

with the vertical length scale σ_f and the characteristic length scale ℓ. In most cases, the exponent γ is set to one or two for the exponential or squared exponential kernel, respectively. Fig. 2.3 shows the kernel function for varied parameters γ and ℓ while σ_f is set to one. As expected, the covariance of two input locations gets higher when points are close together and has its maximum for $x_i = x_j$. With an increasing characteristic length scale ℓ, the kernel function becomes wider which means that random variables are more correlated which results in smoother functions. On the other hand, a small characteristic length scale can be used to represent less smooth functions.



(a) $\gamma = 1$ (b) $\gamma = 2$

Figure 2.3: Different γ-exponential kernel functions for variations of the characteristic length scale ℓ.

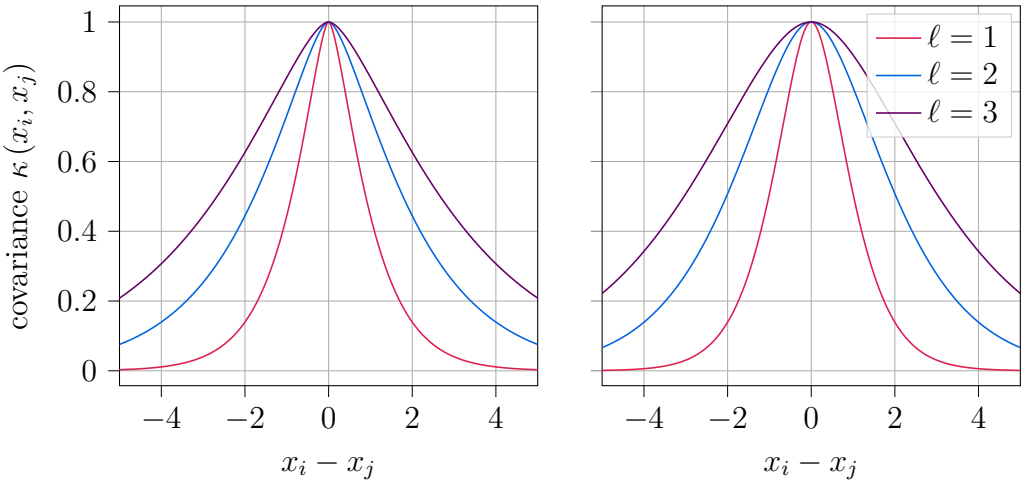
The Matérn Kernel

Another class of covariance functions are the Matérn covariance functions which also include the γ-exponential functions and are thus a generalization. This kernel function was first presented in [Matérn60] and compared to a variety of other kernel functions in [KangEtAl17]. In [KangEtAl17] promising results have been obtained using a Matérn covariance function. According to [RasmussenWilliams06],

the Matérn covariance function is defined as

$$\kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \frac{2^{1-\nu}}{\Gamma\left(\nu\right)} \left(\frac{\sqrt{2\nu}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\ell} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\ell} \right) \tag{2.26}$$

where $\nu > 0$ is the order, $\Gamma\left(\nu\right)$ is the gamma function, K_ν is the modified Bessel function of the second kind and ℓ is the characteristic length scale. Note, that for $\nu = 0.5$ this kernel results in the exponential kernel and for $\nu \to \infty$ the resulting kernel is the squared exponential function. An illustration of the Matérn covariance functions is given in Fig. 2.4.



(a) $\nu = 1$ (b) $\nu = 2$

Figure 2.4: Different kernel functions of the Matérn family for variations of the characteristic length scale ℓ.

2.3.2 Gaussian Process Regression (GPR)

Having defined the full GP including its kernel function, it can be used as a regression tool for an unknown function by conditioning the multivariate Gaussian density of the function values $\boldsymbol{f}$ on collected measurements y_i. For this purpose, an initial mean and covariance matrix have to be defined which can include some prior knowledge. As reported by [RasmussenWilliams06], however, a zero mean function is sufficient in most cases.

Subsequently, the notion of a *training set* $\mathcal{D}_{train}$ and a *test set* $\mathcal{D}_{test}$ is used. The test set

$$\mathcal{D}_{test} = \left\{ \left(\boldsymbol{x}_1, f_1\right), \left(\boldsymbol{x}_2, f_2\right), \ldots, \left(\boldsymbol{x}_n, f_n\right) \right\} \tag{2.27}$$

contains n tuples of input locations $\boldsymbol{x}_i$ where one wants to infer the corresponding field values f_i which are represented by the multivariate Gaussian distribution $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x}))$. The set of test locations is also defined as $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$, where d is the input dimension, e.g. $d = 2$ for a two-dimensional field. The training set

$$\mathcal{D}_{train} = \{(\boldsymbol{x}_1^*, y_1), (\boldsymbol{x}_2^*, y_2), \ldots, (\boldsymbol{x}_n^*, y_m)\} \tag{2.28}$$

includes all collected measurements $\boldsymbol{f}_y = [y_1, y_2, \ldots, y_m]^T \in \mathbb{R}^m$ at the measurement locations $\boldsymbol{X}_* = [\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \ldots, \boldsymbol{x}_n^*]^T \in \mathbb{R}^{m \times d}$. The goal of Gaussian Process Regression (GPR) is to improve the predicted values of the test set by conditioning them on the training set.

Noise Free Observations

First, noise free observations are examined. In this case, the collected measurements are the true field values, reading

$$y_i = f(\boldsymbol{x}_i^*). \tag{2.29}$$

In order to condition the Gaussian PDF of field values $\boldsymbol{f}$ on collected measurements $\boldsymbol{f}_y$, the joint Gaussian distribution of the form

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{bmatrix} \right) \tag{2.30}$$

needs to be defined. Here, the joint covariance matrix can be assembled using the covariance function κ, yielding

$$\boldsymbol{K} = \kappa(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n \times n} \tag{2.31}$$

$$\boldsymbol{K}_* = \kappa(\boldsymbol{X}, \boldsymbol{X}_*) \in \mathbb{R}^{n \times m} \tag{2.32}$$

$$\boldsymbol{K}_{**} = \kappa(\boldsymbol{X}_*, \boldsymbol{X}_*) \in \mathbb{R}^{m \times m}. \tag{2.33}$$

Following the results of Eq. (2.11), the conditional distribution of the predicted field values, given the observations, results in

$$p\left(\boldsymbol{f} | \boldsymbol{f}_y\right) = \mathcal{N}\left(\boldsymbol{f}; \boldsymbol{\mu}_{\boldsymbol{f}|\boldsymbol{f}_y}, \boldsymbol{\Sigma}_{\boldsymbol{f}|\boldsymbol{f}_y}\right) \tag{2.34}$$

with the conditional moments

$$\boldsymbol{\mu}_{\boldsymbol{f}|\boldsymbol{f}_y} = \boldsymbol{\mu} + \boldsymbol{K}_*^T \boldsymbol{K}_{**}^{-1} \left(\boldsymbol{f}_y - \boldsymbol{\mu}_*\right) \tag{2.35}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{f}|\boldsymbol{f}_y} = \boldsymbol{K} - \boldsymbol{K}_*^T \boldsymbol{K}_{**}^{-1} \boldsymbol{K}_*. \tag{2.36}$$

Fig. 2.5 shows the prior of a one dimensional GP with a squared exponential covariance function and a test set of 100 equidistantly distributed points. The colored lines represent sampled functions from the prior distribution while the mean function is initialized to zero. The smoothness of the sampled curves is induced by the hyper-parameters σ_f and ℓ of the covariance function.

The conditioning step is illustrated in Fig. 2.6. The training set contains five exact measurements of the true function which results in zero variance at the measurement locations. It can be seen that the mean function of the posterior distribution $p\left(\boldsymbol{f}|\boldsymbol{f}_y\right)$ roughly matches the true function in areas where dense training data is available and remains close to the prior where little data is available.

Noisy Observations

The assumption of noise free observations does not hold in real world applications. In most cases, the measurement is corrupted by noise which is often modeled to be Gaussian distributed. Instead of the exact measurement in Eq. (2.29), the observation is modeled as

$$y_i = f\left(\boldsymbol{x}_i^*\right) + \varepsilon \tag{2.37}$$

where $\varepsilon \sim \mathcal{N}\left(0, \sigma_y^2\right)$ is a scalar random variable with measurement variance σ_y^2. In contrast to e.g. the vertical length scale σ_f in Eq. (2.25) this variance is not a hyper-parameter and has to be identified in advance. With the aim of incorporating this modification in the GP, the covariance matrix of the measurements

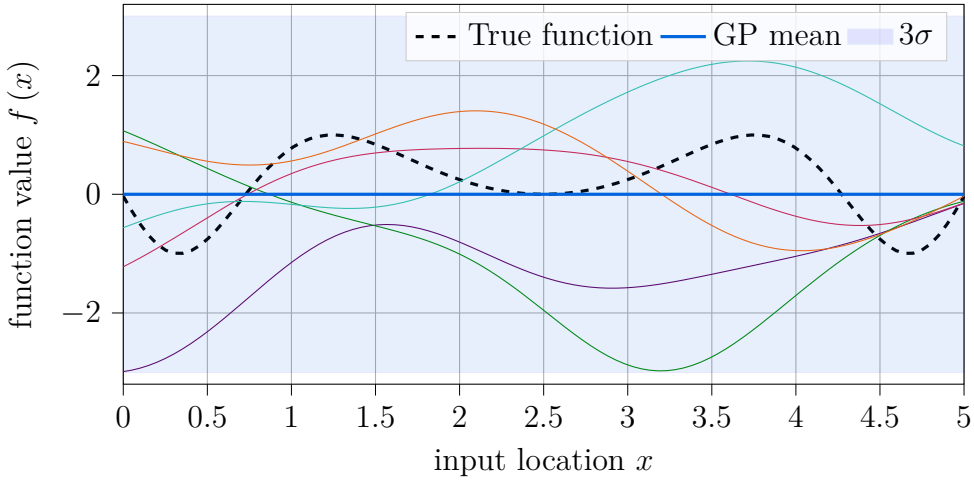


Figure 2.5: Sampled functions from the prior distribution $p\left(\boldsymbol{f}\right)$.

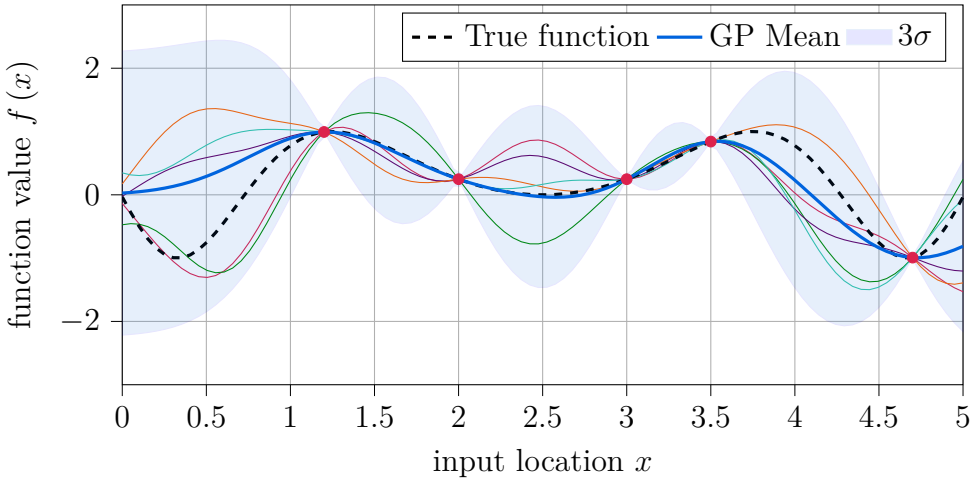


Figure 2.6: Sampled functions from the posterior distribution $p\left(\boldsymbol{f}|\boldsymbol{f}_y\right)$ with measurements indicated by (•).

$\boldsymbol{K}_{**}$ has to include the measurement variance, reading

$$\boldsymbol{K}_{**} = \kappa\left(\boldsymbol{X}_{**}, \boldsymbol{X}_{**}\right) + \sigma_y^2 \boldsymbol{I}, \tag{2.38}$$

where $\boldsymbol{I} \in \mathbb{R}^{m \times m}$ is the identity matrix. Substituting Eq. (2.38) in Eq. (2.35) yields the conditional moments for a noise corrupted GP. Figure 2.7 shows the posterior distribution of a GP with noisy observations. The main difference to the noise free GP can be seen at the measurement locations. Instead of zero uncertainty, the variance is reduced to the measurement variance which results in small deviations from the true function at the sampling locations.

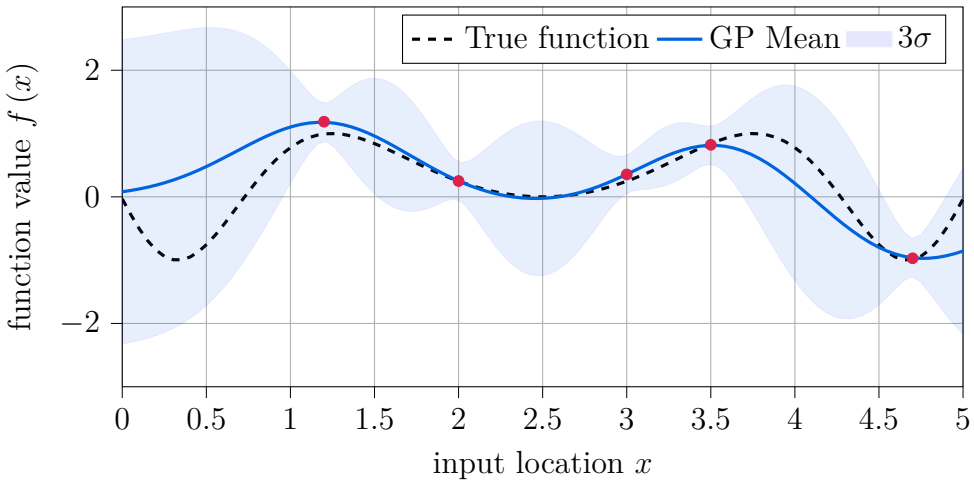


Figure 2.7: Posterior distribution $p\left(\boldsymbol{f}|\boldsymbol{f}_y\right)$ with noise corrupted measurements indicated by (•).

2.4 Weighted Shape Functions

Oftentimes, a continuous space of interest $\mathcal{F} \subseteq \mathbb{R}^d$ of dimension d is discretized into a finite set of gridpoints $\boldsymbol{s}_k \in \mathcal{S}$ where $\mathcal{S} = (\mathcal{V}, \mathcal{E})$ defines a grid consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. This discretization has its probably most popular application in the Finite Element Method (FEM) where a continuum is discretized with a number of so called finite elements. In the two-dimensional case, these finite elements are typically enclosed by three or four vertices depending on the application. Then, a physical property, e.g. temperature, within the finite element can be approximated as a weighted sum of the adjacent vertices.

In this thesis, the field belief representation is also defined on a grid $\mathcal{S}$ to reduce the computational complexity and enable online field exploration. Since the robot, however, is collecting measurements at locations $\boldsymbol{x}_k \in \mathcal{F}$ in a continuous space, a mapping $\boldsymbol{\phi} : \mathcal{S} \mapsto \mathcal{F}$ has to be defined in order to update the field values at discrete locations. As widely used in the FEM, this mapping can be found by using *weighted shape functions*.

In [KreuzerSolowjow18], polynomial basis functions are proposed to approximate a continuous quantity $\eta(\boldsymbol{x})$ in a sub domain $\mathcal{F}^e \subset \mathcal{F}$ using the values $\eta(\boldsymbol{s}_i)$ at the four adjacent vertices $\boldsymbol{s}_i$, reading

$$\eta(\boldsymbol{x}_k) = \sum_{i=1}^{4} \phi_i^e(\boldsymbol{x}_k)\, \eta(\boldsymbol{s}_i) \tag{2.39}$$

Here, $\phi_i^e(\boldsymbol{x}_k)$ denotes the ith shape function in element domain $\mathcal{F}^e$ evaluated at the continuous location $\boldsymbol{x}_k$. Fig. 2.8 illustrates the concept of a single shape function on an element domain. All locations within an element are expressed in a local coordinate system $\mathcal{K}(r, s)$ which is centered in the element. The weighted shape functions are chosen such that

$$\phi_i^e(\boldsymbol{s}_k) = \begin{cases} 1 & , \quad k = i \\ 0 & , \quad \text{else} \end{cases} \tag{2.40}$$

holds which ensures that the approximated quantity $\eta(\boldsymbol{x}_k)$ matches the exact values at grid locations $\boldsymbol{s}_i$.

Using the element dimensions ℓ_r and ℓ_s, the weighted shape functions are defined as

$$\phi_1^e = -\frac{1}{\ell_r \ell_s}\left(r - \frac{\ell_r}{2}\right)\left(s + \frac{\ell_s}{2}\right), \qquad \phi_2^e = \frac{1}{\ell_r \ell_s}\left(r + \frac{\ell_r}{2}\right)\left(s + \frac{\ell_s}{2}\right) \tag{2.41}$$

$$\phi_3^e = \frac{1}{\ell_r \ell_s}\left(r - \frac{\ell_r}{2}\right)\left(s - \frac{\ell_s}{2}\right), \qquad \phi_4^e = -\frac{1}{\ell_r \ell_s}\left(r + \frac{\ell_r}{2}\right)\left(s - \frac{\ell_s}{2}\right). \tag{2.42}$$

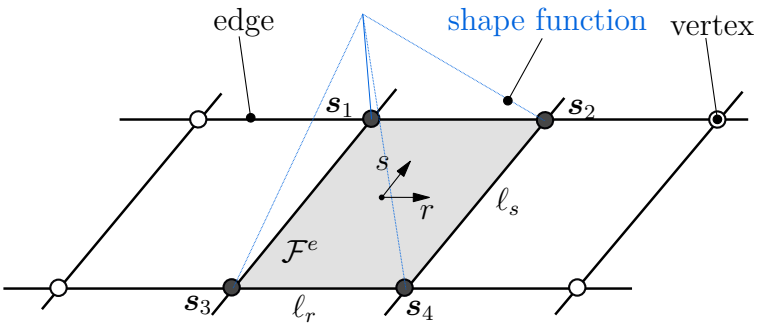


Figure 2.8: Illustration of a single shape function on an element domain $\mathcal{F}^e$ on a regular grid $\mathcal{S}$.

In addition, the continuous mapping in Eq. (2.39) can be rewritten in vector form

$$\eta\left(\boldsymbol{x}_k\right) = \boldsymbol{\phi}^e\left(\boldsymbol{x}_k\right)\boldsymbol{\eta}_s \tag{2.43}$$

where $\boldsymbol{\eta}_s = [\eta_{s_1}, \ldots, \eta_{s_N}]^T \in \mathbb{R}^N$ is the vector of grid values and $\boldsymbol{\phi}^e = [0, \ldots, 0, \phi_1^e, \ldots, \phi_4^e, 0, \ldots, 0] \in \mathbb{R}^{1\times N}$ is the element mapping vector which only contains four non zero elements.

Chapter 3

Field Modeling

This chapter covers the first module of the overall exploration algorithm, namely the field belief representation. Hereby, the main goal is to learn an environmental field from a set of taken measurements which allows for inference of field values at unobserved locations. The previously discussed results on multivariate Gaussian distributions and Gaussian Process Regression (GPR) play an important role. An essential requirement for the field belief description is that it can be updated in an online fashion since the exploring robot is moving through an unknown environment, gathering measurements and planning its path based on the current belief. This means that the field belief needs to be updated sequentially as new information becomes available. Moreover, all computations for updating and storing the field belief have to be carried out on-board as communication to a central computer is not feasible in the underwater domain. As a consequence, the belief algorithm should be limited in computation time in order to be executed on computationally constrained platforms like the HippoCampus.

In the following, the most promising field belief representation is chosen based on a literature research. This algorithm is explained in depth in Section 3.2 and compared to other state of the art field belief models in Section 3.4.

3.1 Related Work

Classical methods for modeling environmental fields rely on first principles of physics. These mathematical models can be expressed in form of a partial differential equation (PDE) which describes the dynamics of a field. In [WiedemannEtAl17] a two-dimensional diffusion model, resulting in a PDE, is used to model the dynamic behavior of gas concentrations over space and time. However, these belief models require solving the PDE numerically which is com-

putationally expensive and thus unsuitable for online applications. Moreover, the quality of the PDEs solution scales with the uncertainty in the physical properties of the process like the gas density. Consequently, this thesis focuses on statistical methods rather than physics based models. These probabilistic belief models rely on observed data and provide a measure of uncertainty in the belief which is crucial for the informative path planning algorithm presented in Chapter 4.

The previously presented GPs are commonly used in geo-statistics to represent environmental fields [Krige51]. Using GPs as field belief description is of much interest since it allows for a continuous field representation including the mean belief and a covariance function. For the special scenario of underwater exploration, however, GPs are computationally too time consuming because of their increasing computational complexity over time.

A widely used method which is mainly applied to gas distribution mapping (GDM) is the Kernel DM+V algorithm presented in [LilienthalEtAl09] and extended to dynamic fields in [ReggenteLilienthal10]. The Kernel DM+V is a non-parametric approach which treats gas distribution modeling as a density estimation problem. In order to create a map of gas distributions for inference, the operating space is discretized into grid cells and a squared exponential kernel is used to model the spatial dependency between grid cells. Then, a mean value and variance are assigned to each grid cell resulting in a field representation. Although this algorithm works well for GDM, it has one major drawback: Building the variance map has a computational complexity which is linear in the number of measurements. This is clearly not suitable for a robot which is gathering information sequentially since the computation time increases with the number of measurements and can become intractable.

Another approach has been presented in [BlancoEtAl13]. The authors propose a field belief description in form of a Kalman Filter to solve the GDM problem where the space is also discretized into grid cells. Random variables are defined for each grid cell which are then collected in a state that represents the field belief. In this special case, the belief is updated using only a correction step, hence there is no underlying process model. Besides from the mean values of the field belief, this approach also offers an uncertainty measure since the state covariance matrix is updated in each correction step. This belief model can be updated sequentially in constant time and is thus applicable to online field exploration. However, the main problem of Kalman Filters is their scalability since the computational cost for updating the belief is high for large state dimensions. Because of that, the authors of [BlancoEtAl13] state that this approach can only be used to build small maps of a gas distribution.

In [StachnissEtAl09], sparse Gaussian process mixture models are used to model gas distributions. In this case the overall field representation is a weighted sum of

individual GPs. A new GP is created whenever the previous GP is not capable to model the gas distribution sufficiently well. Then, each data point is assigned with a probability that it corresponds to a certain GP. This probability is iteratively learned via expectation maximization (EM). This method outperforms a single GP but has yet only been used offline.

A recently proposed method for representing environmental fields are *Gaussian Markov Random Fields* (GMRFs). This method was first presented by [LindgrenRueLindström11] and used as field representation for an underwater field exploration in [KreuzerSolowjow18] and [DueckerEtAl19]. A GMRF is roughly speaking a finite-dimensional multivariate Gaussian distribution and thus an approximation of a GP. Furthermore, as a consequence of the Markov property, the information matrix of the multivariate PDF is sparsely populated which enables fast computations of matrix manipulations as well as improved scalability to higher dimensions. The belief model can be updated sequentially in constant time which is based on Bayesian inference. As it has already been shown in [Mersch19] and [Geist18] that GMRFs are appropriate for online mapping of environmental fields, the main focus of this work is on GMRFs.

3.2 Gaussian Markov Random Fields

In many areas of application, especially in geo-statistics, GPs are used to represent environmental fields. GPs are highly suitable for modeling environmental fields, since the field values can be modeled in continuous space without any discretizations using an infinite dimensional multivariate Gaussian distribution, reading

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)). \tag{3.1}$$

Here, f is the field value and $\boldsymbol{x} \in \mathbb{R}^d$ denotes the spatial location. For dimensions $d > 1$, GPs are also referred to as Gaussian Fields (GFs) which is used subsequently.

Using GFs for an autonomous exploration scenario, however, brings some problems with it. Consider the scenario where a robot is collecting measurements sequentially and updates its field belief every time a new measurement is available. First, the computation time increases with the number of measurements since the posterior distribution of the GP is obtained by conditioning the field values of the test set on the entire training set which grows with the number of taken measurements. The computation time of conditioning the multivariate Gaussian distribution is mainly driven by inverting the covariance matrix of measurements $\boldsymbol{K}_{**} \in \mathbb{R}^{m \times m}$ in Eq. (2.11) which has a computational complexity of $\mathcal{O}(m^3)$. Therefore, the computational complexity of a sequentially updated

GF scales cubically with the number of taken measurements which is often referred to as the *big n problem* [JonaGianluca13]. For an autonomous robot with limited hardware, the conditioning step can become intractable with increasing exploration time. Secondly, all taken measurements and their corresponding locations have to be stored in memory which is unsuited for platforms with limited computational memory.

In this work, the *big n problem* is overcome using Gaussian Markov Random Fields (GMRFs) which approximate the GF on a grid of fixed dimension. This enables efficient computation and storage of the environmental field values. Next, the theory of GMRFs, presented in [LindgrenRueLindström11], is reviewed and compared to other state of the art belief algorithms.

3.2.1 GMRF Construction

While GPs are defined in continuous space, GMRFs are represented by a finite number of random variables defined on a fixed lattice which form a joint Gaussian PDF. Furthermore, GMRFs incorporate the Markov assumption which states that a random variable can be estimated with knowledge of neighboring variables. As a consequence, the information matrix $\mathbf{\Omega}$ of the joint PDF is sparsely populated which allows for fast computations. According to [LindgrenRueLindström11], the construction of GMRFs can be separated into three individual steps:

1. Model the environmental field as a GF on a finite set of locations $\mathcal{S}$, to construct a discretized GF with covariance matrix $\mathbf{\Sigma}$
2. Find a GMRF with precision matrix $\mathbf{\Omega}$ which represents the GF such that $\mathbf{\Omega}$ is close to $\mathbf{\Sigma}^{-1}$ in some norm
3. Use the GMRF representation for the computations using fast numerical methods for sparse matrices

In the following, these three steps are reviewed and a sequential Bayesian update based on [ElizabethEtAl12] and [DueckerEtAl19] for GMRFs is presented. For simplicity, all of the following results are restricted to two-dimensional fields.

Discretizing the Gaussian Field

In this section, a lattice representation of the field is derived. For that purpose, the continuous region of interest $\mathcal{F} \subseteq \mathbb{R}^2$ is discretized into a grid $\mathcal{S} = (\mathcal{V}, \mathcal{E})$ consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. The vertices $\mathcal{V} := \{1, \dots, n\}$ represent the random variables f_i which model the environmental field values $f(\boldsymbol{s}_i)$ at fixed

locations $\boldsymbol{s}_i \in \mathcal{F}$, as it can be seen in Fig. 3.1. Each random variable defines a marginal distribution $p\left(f_i; \mu_i, \sigma_{ii}\right)$ at a spatial location $\boldsymbol{s}_i$.

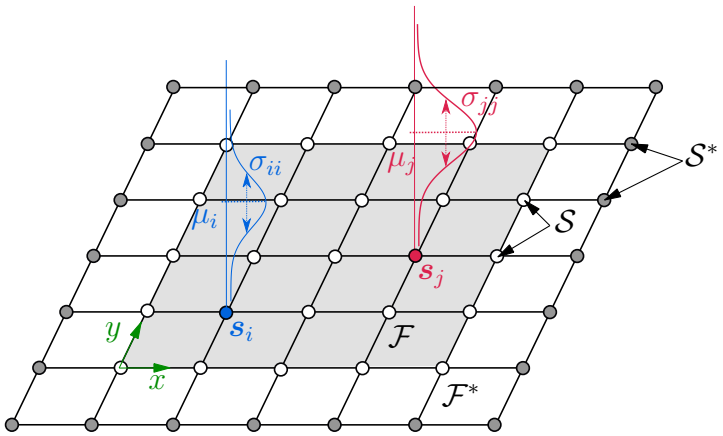


Figure 3.1: Illustration of the discretization of the continuous space $\mathcal{F}$ into a grid $\mathcal{S}$. $\mathcal{F}^*$ and $\mathcal{S}^*$ represent extensions of the area of interest which are needed for boundary conditions.

Then, the discrete Gaussian field

$$f\left(\boldsymbol{x}\right) \sim \mathcal{GP}\left(\mu\left(\boldsymbol{x}\right), \kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\right) \tag{3.2}$$

can be obtained by forming the joint Gaussian distribution of random variables $\boldsymbol{f} = \left[f_1, f_2, \ldots, f_n\right]^T \sim \mathcal{N}\left(\boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f\right)$ where n is the total number of random variables. The covariance matrix $\boldsymbol{\Sigma}_f$ can be obtained using a kernel function, yielding

$$\boldsymbol{\Sigma}_f = \kappa\left(\boldsymbol{S}, \boldsymbol{S}\right) \tag{3.3}$$

where $\boldsymbol{S} = \left[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\right]$ contains the spatial locations of the vertices. As stated in section 2.3, a GP is fully defined by its mean function $\mu\left(\boldsymbol{x}\right)$ and its covariance function $\kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$. The authors of [LindgrenRueLindström11] propose that, in order to represent a GF as GMRF, a covariance function of the Matérn family should be used. The resulting Gaussian field which is also called Matérn field is the solution of a specific stochastic PDE which is frequently used to model environmental processes. Therefore, the Matérn covariance function of order ν, introduced in Eq. (2.26), is used to model the spatial dependencies of random variables f_i and f_j.

Furthermore, as stated in [LindgrenRueLindström11], an important requirement for a GMRF representation of a GF is that it should offer a considerable computation speedup in comparison to a standard GF representation. This can be achieved by using the Markov assumption which is presented in the following section.

Incorporating the Markov Assumption

A fundamental concept in probability is *conditional independence* which is not to be confused with absolute independence. Two random variables f_i and f_j are said to be conditionally independent if f_j carries no information about f_i if the values of the remaining variables $\boldsymbol{f}_{-ij}$ are known. This can be equivalently stated as

$$p\left(f_i|f_j, \boldsymbol{f}_{-ij}\right) = p\left(f_i|\boldsymbol{f}_{-ij}\right) \Leftrightarrow f_i \perp\!\!\!\perp f_j|\boldsymbol{f}_{-ij} \Leftrightarrow \Omega_{ij} = 0, \quad \forall i \neq j \tag{3.4}$$

where Ω_{ij} is the corresponding entry of the information matrix. For a complete derivation of Eq. (3.4), the reader is referred to [RueHeld05]. Especially the result that $\Omega_{ij} = 0$ holds for conditionally independent random variables is of particular importance since this property is used in the following to construct a sparse information matrix. In the case of GMRFs, conditional independence can be imposed using the *Markov assumption* which states that a random variable f_i can be inferred by the knowledge of its grid neighbor variables $\boldsymbol{f}_{n(i)}$ which are connected by an edge. An illustration of the Markov assumption can be seen in Fig. 3.2 where the dark grey vertex in the center is conditionally independent of all non-connected white vertices, given the neighbor vertices which are colored in light grey. The Markov assumption is therefore a relaxation of the definition of

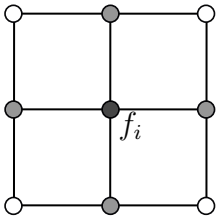


Figure 3.2: Illustration of the Markov assumption on a two-dimensional grid. The random variable f_i (dark grey) is conditionally independent of all non-connected random variables (white) given its neighbors values $\boldsymbol{f}_{n(i)}$ (light grey).

conditional independence in Eq. (3.4). In this case, two random variables f_i and f_j are considered to be conditionally independent if the values of the neighboring random variables $\boldsymbol{f}_{n(i)}$ are known, hence

$$p\left(f_i|f_j, \boldsymbol{f}_{n(i)}\right) = p\left(f_i|\boldsymbol{f}_{n(i)}\right) \quad \forall j \notin \{n(i), i\}\,. \tag{3.5}$$

Therefore, the random variable f_i is fully defined by the knowledge of its neighbor variables.

Incorporating the Markov assumption in the discretized Gaussian field, presented in the previous section, restricts the entries of the information matrix of the

multivariate Gaussian PDF to

$$\Omega_{ij} \neq 0 \Leftrightarrow \{i,j\} \in \mathcal{E}, \quad \forall i \neq j, \tag{3.6}$$

which states that the information of two random variables is non zero if and only if they are connected by an edge. If the edge structure of the grid is chosen in a way that a random variable is only conditionally dependent of its k adjacent neighbors, the information matrix $\boldsymbol{\Omega}$ results in a sparse matrix. This is of course a simplification, since it is not ensured that a random variable is conditionally independent of all remaining variables except for its neighbors. However, this is the fundamental trade-off between accuracy of the field representation and computational complexity and thus it is a modeling assumption.

To summarize the previous steps, a GMRF is an approximation of a GF and is defined by a vector of random variables $\boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}\right)$ which is assumed to be Gaussian distributed with information matrix $\boldsymbol{\Omega} > 0$ which is defined on the grid $\mathcal{S}$ and restricted to Eq. (3.6). Defining a mean $\boldsymbol{\mu}$ and an information matrix $\boldsymbol{\Omega}$ is therefore sufficient to fully define a GMRF. According to [RueHeld05], the Gaussian full conditionals of a GMRF are given by

$$\mathbb{E}\left(f_i|\boldsymbol{f}_{-i}\right) = \mu_i - \frac{1}{\Omega_{ii}} \sum_{j,j\neq i}^{n} \Omega_{ij}\left(f_j - \mu_j\right) \tag{3.7}$$

$$Prec\left(f_i|\boldsymbol{f}_{-i}\right) = \Omega_{ii} \tag{3.8}$$

$$Corr\left(f_i, f_j|\boldsymbol{f}_{-ij}\right) = -\frac{\Omega_{ij}}{\sqrt{\Omega_{ii}\Omega_{jj}}}, \quad \forall i \neq j \tag{3.9}$$

which relate the conditional expectation $\mathbb{E}\left(f_i|\boldsymbol{f}_{-i}\right)$, the conditional precision $Prec\left(f_i|\boldsymbol{f}_{-i}\right)$ and the conditional correlations $Corr\left(f_i, f_j|\boldsymbol{f}_{-ij}\right)$ to the entries of the mean vector and the information matrix. As expected, the conditional expectation of a random variable f_i is a weighted sum of the conditionally dependent variables f_j since variables with $\Omega_{ij} = 0$ have no influence.

In this work, the grid structure is chosen, as illustrated in Fig. 3.1 and Fig. 3.2, where a grid value is only influenced by its direct neighbors. As a consequence, the Gaussian full conditionals are only dependent of the four adjacent random variables $\boldsymbol{f}_{n(i)}$. Such conditionals are also known as Conditional Autoregressive processes (CAR) of order one, [Besag74]. Extensions, where the next k nearest neighbors are taken into consideration are called CAR(k) processes. These have been intensively investigated in [Geist18] and are therefore not covered in this work. A GMRF, defined as a CAR(1) model, is chosen in this thesis because of its minimal computational complexity. Furthermore, according to [LindgrenRueLindström11], a CAR(1) GMRF is an approximation of a GF with Matérn covariance function for $\nu \to 0$ and is thus an approximate solution to many environmental processes.

Assuming a zero initial mean, the full conditionals of a CAR(1) process read

$$\mathbb{E}\left(f_i|\boldsymbol{f}_{-i}\right) = \frac{1}{a}\sum_{k\in n(i)} f_k = \frac{1}{a}\left(f_{lower} + f_{upper} + f_{left} + f_{right}\right) \tag{3.10}$$

$$Prec\left(f_i|\boldsymbol{f}_{-i}\right) = a \tag{3.11}$$

$$\tag{3.12}$$

where $|a| > 4$ is required to ensure that $\boldsymbol{\Omega}$ stays positive definite. Accordingly, the entries of the information matrix are given by

$$\Omega_{ii} = a \quad \forall i : \boldsymbol{s}_i \in \mathcal{S} \tag{3.13}$$

$$\Omega_{ij} = -1 \quad \forall \{i,j\} \in \mathcal{E}, i \neq j. \tag{3.14}$$

The entire derivation that Equations 3.10 to 3.14 are indeed an approximation of a GF is omitted for the readers convenience and can be found in [LindgrenRueLindström11].

As it can be noticed, the formulation of the full conditionals does not hold at the boundaries of the grid $\mathcal{S}$ since at least one variable is not available. To overcome that problem, the grid $\mathcal{S}$ defined on the region of interest $\mathcal{F}$ is extended to a grid representation $\mathcal{S}^*$ defined on $\mathcal{F}^*$ by adding vertices to each boundary, as it can be seen in Fig. 3.1. In [Geist18], Neumann boundary conditions are used which model the environmental field as constant outside of $\mathcal{F}$ meaning that it has the same value as the predicted field value at the boundary. For example, the conditional expectation of the upper left corner of the field with coordinates $x_c = 0$ and $y_c = y_{max}$ results in

$$\mathbb{E}\left(f_i|\boldsymbol{f}_{-i}\right) = \frac{1}{a}\sum_{k\in n(i)} f_k = \frac{1}{a}\left(f_{lower} + f_{right} + 2\mathbb{E}\left(f_i|\boldsymbol{f}_{-i}\right)\right) \tag{3.15}$$

$$= \frac{1}{a-2}\left(f_{lower} + f_{right}\right) \tag{3.16}$$

where the neighboring variables f_{upper} and f_{left} are substituted by $\mathbb{E}\left(f_i|\boldsymbol{f}_{-i}\right)$. Incorporating the boundary conditions in the information matrix can be achieved by setting $\Omega_{ii} = a - 2$ in corner vertices and $\Omega_{ii} = a - 1$ at wall vertices, respectively.

So far, the full GMRF representation of a GF has been derived. Next, conditioning of GMRF variables is covered to update the field belief based on observations of the true field.

3.2.2 Sequential Conditioning of GMRFs

The goal of this section is to provide a formalism for updating the GMRF belief consisting of the mean vector $\boldsymbol{\mu}$ and the information matrix $\boldsymbol{\Omega}$, as new measure-

ments are available. First, weighted shape functions are reviewed in order to update the GMRF variables at discrete locations based on measurements from a continuous domain $\mathcal{F}$. Then, the concepts of multivariate Gaussian distributions, presented in Chapter 2, are used to derive a Bayesian sequential update of the GMRF which enables recursive state estimation.

To approximate field values at continuous locations which are not directly at the discrete vertex locations, a mapping $\boldsymbol{\Phi} : \mathcal{S} \mapsto \mathcal{F}$ is needed. As specified in Section 2.4, this mapping can be defined using weighted shape functions which make it possible to obtain an approximated marginal distribution of a field value $f_{\boldsymbol{x}}$ at an arbitrary position $\boldsymbol{x} \in \mathcal{F}$ that is not located on the grid. This marginal PDF reads

$$p\left(f_{\boldsymbol{x}}\right) = \mathcal{N}\left(f_{\boldsymbol{x}}; \mu_{\boldsymbol{x}}, \sigma^2_{\boldsymbol{x}}\right) \tag{3.17}$$

with mean and variance

$$\mu_{\boldsymbol{x}} = \boldsymbol{\phi}\left(\boldsymbol{x}\right) \boldsymbol{\mu} \tag{3.18}$$

$$\sigma^2_{\boldsymbol{x}} = \boldsymbol{\phi}\left(\boldsymbol{x}\right) \operatorname{diag}\left(\boldsymbol{\Sigma}\right). \tag{3.19}$$

Here, the mean at a continuous location $\mu_{\boldsymbol{x}}$ is a weighted sum of the four neighboring mean concentrations which is calculated using the mapping vector $\boldsymbol{\phi}\left(\boldsymbol{x}\right) = \left[0, \ldots, 0, \phi^e_1\left(\boldsymbol{x}\right), \ldots, \phi^e_4\left(\boldsymbol{x}\right), 0, \ldots, 0\right] \in \mathbb{R}^{1\times n}$, defined in Eq. (2.43). Similarly, the variance is obtained using the marginal variances of adjacent vertices which are stored in $\operatorname{diag}\left(\boldsymbol{\Sigma}\right)$. Especially, the variance at a continuous location $\sigma^2_{\boldsymbol{x}}$ is of much interest since the path planning algorithm, covered in Chapter 4, evaluates potential measurement locations in continuous space based on their predictive variance.

Furthermore, the measurement probability, relating the GMRF belief $\boldsymbol{f}$ to a single measurement y_t taken at location $\boldsymbol{x}_t$, can be defined as

$$p\left(y_t|\boldsymbol{f}\right) = \boldsymbol{\phi}_t\left(\boldsymbol{x}_t\right) \boldsymbol{f} + \varepsilon \tag{3.20}$$

with the random variable $\varepsilon \sim \mathcal{N}\left(0, \sigma^2_y\right)$ which is modeled as white Gaussian noise with variance σ^2_y. Accordingly, the measurement probability is also a Gaussian PDF of the form $\mathcal{N}\left(y_t; \boldsymbol{\phi}_t \boldsymbol{f}, \sigma^2_y\right)$. The mapping vector $\boldsymbol{\phi}_t$ contains the four weighted shape functions for the corresponding location $\boldsymbol{x}_t$ where measurement y_t has been taken.

Having defined the measurement probability, an update rule for the posterior distribution $p\left(\boldsymbol{f}|\boldsymbol{y}\right)$ is derived. For a better understanding of the recursive formulation, the notion of a belief, frequently used in [ThrunBurgardFox05], is introduced. The belief at time t

$$bel_t\left(\boldsymbol{f}\right) = p\left(\boldsymbol{f}|\boldsymbol{y}\right) = p\left(\boldsymbol{f}|y_1, y_2, \ldots, y_t\right) \tag{3.21}$$

describes the posterior distribution of field values $\boldsymbol{f}$ given all measurements up to y_t and is characterized by its mean $\boldsymbol{\mu}_t$ and its information $\boldsymbol{\Omega}_t$. Eq. (3.21) summarizes the goal of this section: At every instance in time t, the current belief, hence the posterior distribution of field values, is to be estimated. Using Bayes' rule, this estimation process can be rewritten in recursive form, reading

$$
\begin{aligned}
bel_t\left(\boldsymbol{f}\right) &= p\left(\boldsymbol{f}|y_1, y_2, \ldots, y_t\right) && (3.22)\\
&\propto p\left(y_t|\boldsymbol{f}, y_1, \ldots, y_{t-1}\right) p\left(\boldsymbol{f}|y_t, \ldots, y_{t-1}\right) && (3.23)\\
&= p\left(y_t|\boldsymbol{f}\right) bel_{t-1}\left(\boldsymbol{f}\right). && (3.24)
\end{aligned}
$$

First, to get from Eq. (3.22) to Eq. (3.23), Baye's rule, presented in section 2.2, is applied to relate the current belief to the measurement probability $p\left(y_t|\boldsymbol{f}, y_1, \ldots, y_{t-1}\right)$ given all measurements up to time $t-1$. Since the measurement probability, by definition, only depends on the field belief $\boldsymbol{f}$, the most recent measurement y_t is conditionally independent of all previous measurements $y_1, \ldots, y_{t-1}$ given $\boldsymbol{f}$, resulting in Eq. (3.24). The second term in Eq. (3.23), resulting from Bayes' rule, is the posterior distribution of field values given all measurements up to time $t-1$ and thus can be substituted by the previous belief $bel_{t-1}\left(\boldsymbol{f}\right)$, yielding a recursion for updating the GMRF variables.

As both, the measurement probability as well as the field belief, are represented by multivariate Gaussian PDFs, an update for the canonical parameters of a Gaussian can be derived, yielding

$$
\begin{aligned}
bel_t\left(\boldsymbol{f}\right) &\propto p\left(y_t|\boldsymbol{f}\right) bel_{t-1}\left(\boldsymbol{f}\right) && (3.25)\\
&\propto \exp\left(-\frac{1}{2\sigma_y^2}\left(y_t - \boldsymbol{\phi}_t\boldsymbol{f}\right)^T\left(y_t - \boldsymbol{\phi}_t\boldsymbol{f}\right) - \frac{1}{2}\left(\boldsymbol{f} - \boldsymbol{\mu}_{t-1}\right)^T\boldsymbol{\Omega}_{t-1}\left(\boldsymbol{f} - \boldsymbol{\mu}_{t-1}\right)\right) && (3.26)\\
&\propto \exp\left(-\frac{1}{2}\boldsymbol{f}^T\underbrace{\left(\boldsymbol{\Omega}_{t-1} + \frac{1}{\sigma_y^2}\boldsymbol{\phi}_t^T\boldsymbol{\phi}_t\right)}_{\boldsymbol{\Omega}_t}\boldsymbol{f} + \boldsymbol{f}^T\underbrace{\left(\underbrace{\boldsymbol{\Omega}_{t-1}\boldsymbol{\mu}_{t-1}}_{\boldsymbol{\xi}_{t-1}} + \frac{1}{\sigma_y^2}\boldsymbol{\phi}_t^T y_t\right)}_{\boldsymbol{\xi}_t}\right). && (3.27)
\end{aligned}
$$

First, the moment parametrization of a Gaussian is used for both PDFs resulting in Eq. (3.26) with the prior moments $\boldsymbol{\mu}_{t-1}$ and $\boldsymbol{\Sigma}_{t-1} = \boldsymbol{\Omega}_{t-1}^{-1}$. Then, by multiplying all quantities and rearranging them into the canonical form, the Gaussian PDF of the field belief at time t in Eq. (3.27) can be obtained. Note, that this only provides an update for the canonical parameters $\boldsymbol{\xi}_t$ and $\boldsymbol{\Omega}_t$ which are in general

hard to interpret. However, by first updating the canonical parameters

$$\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} + \frac{1}{\sigma_y^2}\boldsymbol{\phi}_t^T\boldsymbol{\phi}_t \tag{3.28}$$

$$\boldsymbol{\xi}_t = \boldsymbol{\xi}_{t-1} + \frac{1}{\sigma_y^2}\boldsymbol{\phi}_t^T y_t, \tag{3.29}$$

the conditional mean at time t can be obtained by solving the system of equations $\boldsymbol{\Omega}_t\boldsymbol{\mu}_t = \boldsymbol{\xi}_t$. In order to avoid inverting the information matrix, the covariance matrix can be updated sequentially by application of the *Sherman-Morrison* formula

$$\boldsymbol{\Sigma}_t = \left(\boldsymbol{\Omega}_{t-1} + \frac{1}{\sigma_y^2}\boldsymbol{\phi}_t^T\boldsymbol{\phi}_t\right)^{-1} \tag{3.30}$$

$$= \boldsymbol{\Omega}_{t-1}^{-1} - \frac{\boldsymbol{\Omega}_{t-1}^{-1}\boldsymbol{\phi}_t^T\boldsymbol{\phi}_t\boldsymbol{\Omega}_{t-1}^{-1}}{\sigma_y^2 + \boldsymbol{\phi}_t\boldsymbol{\Omega}_{t-1}^{-1}\boldsymbol{\phi}_t^T} \tag{3.31}$$

$$= \boldsymbol{\Sigma}_{t-1} - \frac{\boldsymbol{h}_t\boldsymbol{h}_t^T}{\sigma_y^2 + \boldsymbol{\phi}_t\boldsymbol{h}_t}, \tag{3.32}$$

where $\boldsymbol{h}_t \in \mathbb{R}^n$ can be obtained by solving $\boldsymbol{\Omega}_{t-1}\boldsymbol{h}_t = \boldsymbol{\phi}_t$. Since in most cases, the predicted variance Σ_{ii} and not the covariance Σ_{ij} is of interest, it is sufficient to only update and store the diagonal of the covariance matrix, reading

$$\operatorname{diag}(\boldsymbol{\Sigma}_t) = \operatorname{diag}(\boldsymbol{\Sigma}_{t-1}) - \frac{\boldsymbol{h}_t \odot \boldsymbol{h}_t}{\sigma_y^2 + \boldsymbol{\phi}_t\boldsymbol{h}_t}, \tag{3.33}$$

where $\odot$ denotes the element-wise multiplication.

Note, that for updating the conditional mean and the covariance matrix diagonal, two linear systems of equations have to be solved. Due to the sparsity of the information matrix $\boldsymbol{\Omega}$, this can be done using fast numerical methods for sparse systems, resulting in a computational complexity of $\mathcal{O}(1)$ for the sequential Bayesian update, see [ElizabethEtAl12].

3.2.3 Latent Variable Model

In the previous sections, GMRFs have been derived under the assumption that the GMRF variables represent the environmental field values with, e.g., zero initial mean. In [DueckerEtAl19], however, the environmental field values are represented by latent variables $f_i = f(\boldsymbol{x}_i) \in \mathbb{R}$ which can be expressed using a global linear model

$$f_i = \mu_f(\boldsymbol{x}_i, \boldsymbol{\beta}) + \eta_i, \quad \forall i = 1, \ldots, n \tag{3.34}$$

with the mean regression function $\mu_f(\boldsymbol{x}_i, \boldsymbol{\beta})$ and the GMRF variables η_i which model the deviations of the mean function from the true field. These GMRF variables are assumed to have zero initial mean, hence $\boldsymbol{\eta} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Omega}_\eta^{-1}\right)$, where $\boldsymbol{\Omega}_\eta$ has the same structure as presented in Section 3.2.1. Using the GMRF variables to only model the residuals makes it possible to impose some prior knowledge in the mean regression function, e.g. that the field to be estimated has a sinusoidal shape. This prior knowledge can be incorporated using a linear model, reading

$$\mu_f(\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{m}(\boldsymbol{x})^T \boldsymbol{\beta}, \tag{3.35}$$

where $\boldsymbol{m}(\boldsymbol{x}) = [m_1(\boldsymbol{x}), \ldots, m_r(\boldsymbol{x})]^T \in \mathbb{R}^r$ contains the r regression functions $m_i(\boldsymbol{x})$ and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_r]^T \in \mathbb{R}^r$ are the corresponding unknown regression coefficients. Then, the collection of environmental field values at the discrete grid locations $\boldsymbol{s}_i$ can be represented by

$$\boldsymbol{f} = \boldsymbol{M}^T \boldsymbol{\beta} + \boldsymbol{\eta}, \tag{3.36}$$

with $\boldsymbol{M} = [\boldsymbol{m}(\boldsymbol{s}_1), \ldots, \boldsymbol{m}(\boldsymbol{s}_n)] \in \mathbb{R}^{r \times n}$. It is assumed, that the regression coefficients $\boldsymbol{\beta}$ can also be estimated using a GP with zero initial mean and a prior information matrix $\boldsymbol{\Omega}_{\boldsymbol{\beta}}$, hence $\boldsymbol{\beta} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Omega}_{\boldsymbol{\beta}}^{-1}\right)$. The goal of using a latent GMRF representation is to estimate the regression coefficients $\boldsymbol{\beta}$ and the residuals $\boldsymbol{\eta}$ simultaneously. This can be done by definition of the full latent field $\tilde{\boldsymbol{f}} = \left[\boldsymbol{f}^T, \boldsymbol{\beta}^T\right]^T \in \mathbb{R}^{n+r}$ which is given by the joint Gaussian PDF

$$p\left(\tilde{\boldsymbol{f}}\right) = p(\boldsymbol{f}, \boldsymbol{\beta}) = p(\boldsymbol{f}|\boldsymbol{\beta})\, p(\boldsymbol{\beta}) \tag{3.37}$$

$$\propto \exp\left(-\frac{1}{2}\left(\boldsymbol{f} - \boldsymbol{M}^T\boldsymbol{\beta}\right)^T \boldsymbol{\Omega}_\eta \left(\boldsymbol{f} - \boldsymbol{M}^T\boldsymbol{\beta}\right) - \frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{\Omega}_\beta \boldsymbol{\beta}\right) \tag{3.38}$$

$$= \exp\left(-\frac{1}{2}\tilde{\boldsymbol{f}}^T \tilde{\boldsymbol{\Omega}} \tilde{\boldsymbol{f}}\right) \tag{3.39}$$

with the joint information matrix

$$\tilde{\boldsymbol{\Omega}} = \begin{bmatrix} \boldsymbol{\Omega}_\eta & -\boldsymbol{\Omega}_\eta \boldsymbol{M} \\ -\boldsymbol{M}^T \boldsymbol{\Omega}_\eta & \boldsymbol{M}^T \boldsymbol{\Omega}_\eta \boldsymbol{M} + \boldsymbol{\Omega}_\beta \end{bmatrix} \in \mathbb{R}^{(n+r)\times(n+r)}. \tag{3.40}$$

Note, that the latent field $\tilde{\boldsymbol{f}} \sim \mathcal{N}\left(\mathbf{0}, \tilde{\boldsymbol{\Omega}}^{-1}\right)$ with zero prior mean can be estimated using the previously derived update equations with the only modification, that $\boldsymbol{\phi}_t$ has to be extended by r zeros.

In the following simulations in Section 3.4.2, it is assumed, that the latent field can be modeled as the sum of a constant field concentration β and the residuals $\boldsymbol{\eta}$. Therefore, the regression function is chosen as $\boldsymbol{m}(\boldsymbol{x}) = 1$ which results in $\tilde{\boldsymbol{f}} \in \mathbb{R}^{n+1}$. In the beginning where only a few measurements are available, using a constant mean regression function has the effect that the mean values

of all random variables $\boldsymbol{f}$ are close to the measured concentration. Without a latent variable model, mean values of GMRF variables that are far away from the measurement locations would remain close to zero if a prior zero mean is imposed.

3.3 Benchmark Algorithms

In order to justify the selection of a GMRF as field representation in this work, it is benchmarked against two other state of the art field belief algorithms. The first algorithm chosen for comparison, is the Kernel DM+V because of its broad application in the modeling of gas distributions which are also environmental fields. Secondly, the Kalman Filter based field representation presented in [BlancoEtAl13] is analyzed since the Kalman Filter is one of the best studied techniques for Bayesian inference. Both algorithms are shortly covered in the following. In order to make a fair comparison between the different belief models, the benchmark algorithms are modified which is covered in Appendix A.2.

3.3.1 Kernel DM+V

The Kernel DM+V algorithm was first presented in [LilienthalEtAl09] in 2009. Modifications of this algorithm are the time dependent (TD) Kernel DM+V [AsadiEtAl11] and the 3D Kernel DM+V/W [ReggenteLilienthal10] which is used for three dimensional gas distribution mapping (GDM) taking into account airflow information. The main results of [LilienthalEtAl09] are presented in the following. For an in depth explanation, the reader is referred to the original paper.

To create a representation of an environmental field, the authors propose a discretization of the space into $n = n_x n_y$ grid cells where n_x and n_y are the discretizations in x and y, respectively. Each cell (k) is assigned with the two weights

$$\Omega^{(k)} = \sum_{i=0}^{m} \kappa_2 \left(\boldsymbol{x}_i, \boldsymbol{x}^{(k)} \right) \tag{3.41}$$

$$R^{(k)} = \sum_{i=0}^{m} \kappa_2 \left(\boldsymbol{x}_i, \boldsymbol{x}^{(k)} \right) y_i \quad \forall k = 1, \ldots, n, \tag{3.42}$$

where Ω are the integrated importance weights, R integrates the weighted sensor readings y_i and m is the number of measurements. As mentioned before, a squared exponential kernel κ_2 is used to model the spatial dependency between measurement locations $\boldsymbol{x}_i$ and the center of the grid cell $\boldsymbol{x}^{(k)}$. The authors of

[LilienthalEtAl09] state that the integrated weights Ω also provide a confidence measure for the estimates. High values of $\Omega^{(k)}$ originate from dense sampling locations close $\boldsymbol{x}^{(k)}$. The opposite holds for few measurements available nearby the cell center. To formalize this problem a confidence map

$$\alpha^{(k)} = 1 - \exp\left(-\left(\frac{\Omega^{(k)}}{\sigma_\Omega}\right)^2\right) \tag{3.43}$$

is introduced which provides values close to one if a large number of readings is available and is zero if no measurements have been taken. Besides from the hyper-parameters of the kernel function, the scaling parameter σ_Ω is defined which influences the extrapolation. The confidence map α is used for the mean concentration map

$$r^{(k)} = \alpha^{(k)}\frac{R^{(k)}}{\Omega^{(k)}} + (1 - \alpha^{(k)})r_0 \tag{3.44}$$

as trade-off between weighted measurements R/Ω and the mean field concentration r_0 which is the average over all sensor readings. Here, $r^{(k)}$ represents the estimated field value of the kth grid cell.

In addition to the estimation of a mean concentration map, the Kernel DM+V algorithm also includes a parallel estimation process for the field variance. Similarly to the mean concentration map, the variance map v can be obtained using

$$V^{(k)} = \sum_{i=0}^{m} \kappa_2\left(\boldsymbol{x}_i, \boldsymbol{x}^{(k)}\right)(y_i - r^{(k(i))})^2 \tag{3.45}$$

$$v^{(k)} = \alpha^{(k)}\frac{V^{(k)}}{\Omega^{(k)}} + (1 - \alpha^{(k)})v_0, \tag{3.46}$$

where V is the integrated variance map and $r^{(k(i))}$ is the current mean value of the grid cell $(k(i))$ where measurement y_i has been taken. Again, α is used to weigh off the measured variance and the mean field variance v_0 which is obtained by averaging. In contrast to all other map updates, the integrated variance update in Eq. (3.45) requires iterating over all m previous measurements which results in a computational complexity being linear in the number of measurements. Therefore, the variance map is neglected and a different uncertainty metric based on the confidence map $\alpha^{(k)}$ is defined in Appendix A.2.

3.3.2 Kalman Filter Based Field Representation

Similarly to the discretization for the Kernel DM+V algorithm, the authors of [BlancoEtAl13] propose a two-dimensional discretization with $n = n_x n_y$ scalar variables representing the field value at locations (x, y). These random variables are collected in a state which forms a joint Gaussian distribution, reading $\boldsymbol{m} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In contrast to the Kernel DM+V algorithm, this approach is based on Bayesian inference, see Section 2.2.

The Kalman Filter is probably the most prominent and best studied technique for Bayesian inference which allows to incrementally update the state estimate given new observations z, [ThrunBurgardFox05]. A Kalman Filter (KF) representation usually consists of two main components: The *state transition probability* $p(\boldsymbol{m}_t|\boldsymbol{u}_t, \boldsymbol{m}_{t-1})$ which relates the current state $\boldsymbol{m}_t$ to the previous state and a control input $\boldsymbol{u}_t$ and the *measurement probaility* $p(z_t|\boldsymbol{m}_t)$. Both PDFs are assumed to be Gaussian distributions. In [BlancoEtAl13] the state transition probability is assumed to be the identity map and thus, the proposed KF uses only the scalar measurement probability which is given by

$$p(z_t|\boldsymbol{m}_t) = \boldsymbol{H}_t \boldsymbol{\mu}_t + n_t, \tag{3.47}$$

where $\boldsymbol{H}_t$ is the observation model at time t, $\boldsymbol{\mu}_t$ is the current mean state estimate and $n_t \sim \mathcal{N}\left(0, \sigma_y^2\right)$ is additive Gaussian noise. Because of the absence of a state transition model, the update equations for the mean vector and the covariance matrix are given by

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{K}_t \left(z_t - \boldsymbol{H}_t \boldsymbol{\mu}_{t-1}\right) \tag{3.48}$$

$$\boldsymbol{\Sigma}_t = \left(\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t\right) \boldsymbol{\Sigma}_{t-1} \tag{3.49}$$

where $\boldsymbol{K}_t$ is called the *Kalman gain*. The observation model which relates the state space to the observation space reads

$$\boldsymbol{H}_t = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \tag{3.50}$$

$$\boldsymbol{H}_t[k] = \begin{cases} 1 & , \quad k = c \\ 0 & , \quad \text{else} \end{cases} \tag{3.51}$$

where c denotes the cell in which the measurement z_t has been taken. Therefore, the mean measurement prediction $\boldsymbol{H}_t \boldsymbol{\mu}_t$ is simply the current mean belief at the observed cell. Having defined the observation model, the Kalman gain can be obtained through

$$\boldsymbol{K}_t = \boldsymbol{\Sigma}_{t-1} \boldsymbol{H}_t^T \left(\boldsymbol{H}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{H}_t^T + \sigma_y^2\right)^{-1} \tag{3.52}$$

$$= \boldsymbol{\Sigma}_{t-1}^c \frac{1}{\Sigma_{t-1}^{cc} + \sigma_y^2}. \tag{3.53}$$

Here, $\boldsymbol{\Sigma}_{t-1}^{c}$ denotes the cth column of the prior covariance matrix and Σ_{t-1}^{cc} is the prior variance of cell c. Substituting the Kalman gain in Eq. (3.48) yields a recursive formulation for the field belief including the mean estimate $\boldsymbol{\mu}_t$ and an uncertainty measure in form of the covariance matrix $\boldsymbol{\Sigma}_t$.

To improve the performance of the KF, two modifications are proposed in Appendix A.2. First, the observation model $\boldsymbol{H}_t$ is changed such that measurements are mapped to the four adjacent random variables using weighted shape functions. Second, a state transition probability $p\left(\boldsymbol{m}_t|\boldsymbol{m}_{t-1}\right)$ is defined which includes a squared exponential kernel function for extrapolation in unobserved regions. The modifications are analyzed in Appendix A.2.2.

3.4 Comparison of Belief Algorithms

In this section, GMRFs are compared to the benchmark algorithms presented before. All algorithms can be updated sequentially to estimate an environmental field based on taken measurements and are thus suitable for an online application. Many different criteria can be defined to evaluate the performance of a particular algorithm. For a field belief representation which is applied to an underwater exploration, however, the most important criteria are chosen to be the following in descending order.

Accuracy The accuracy is used to evaluate how well the true field can be represented by the belief model. The metric which is used to quantify the accuracy is the Root-Mean-Squared-Error (RMSE) between the true field and the mean field belief, reading

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(\mu_t\left(\boldsymbol{x}_i\right) - f\left(\boldsymbol{x}_i\right)\right)^2}{n}} \tag{3.54}$$

where n is the number of field variables. This accuracy criterion is chosen because the goal of many exploration missions is to create a map of the underlying process which can then be used to, e.g., locate the maximum in that map. The more accurate the map is, the better conclusions about the true process can be drawn.

Scalability Algorithms are compared regarding their scalability which describes how well the algorithms handle an increase in the number of field variables. More field variables are needed to either improve the accuracy on smaller fields or to enable mapping of large fields. This criterion is chosen, since the exploration should not be limited by the field size which influences the computational effort of updating the field belief. The scalability is

rated using the computational complexities of the algorithms as well as the computation time for a belief update.

Uncertainty Measure In order to enable informative path planning, an uncertainty measure is needed in the belief description. Therefore, this criterion is selected to evaluate the quality of the uncertainty belief.

Before diving into simulations, the computational complexities of the algorithms are summarized in the following.

3.4.1 Computational Costs

Updating a belief using Gaussian Process Regression (GPR) is based on conditioning a multivariate Gaussian distribution on all m collected measurements. This conditioning step involves inverting a dense matrix of dimension $\mathbb{R}^{m \times m}$ which has a computational complexity $\mathcal{O}(m^3)$. This is computationally clearly infeasible for an online application since the computation time increases with the number of measurements.

The modified sequential Kernel DM+V algorithm, presented in section 3.3.1, is independent of the number of collected measurements. The computational complexity is mainly driven by the summation of two matrices of dimensions $\mathbb{R}^{n_x \times n_y}$. According to [ValadeEtAl17], this results in a computational complexity of $\mathcal{O}(n)$ where $n = n_x n_y$ is the number of grid cells. Note, that this complexity is only constant because the variance map, proposed by the authors of [LilienthalEtAl09], is neglected.

The unmodified Kalman Filter approach proposed in [BlancoEtAl13] has an overall computational complexity of $\mathcal{O}(n^2)$ where n is the state dimension, hence the number of grid cells. This is due to the update of the covariance matrix in Eq. (3.48). The modified Kalman Filter increases the computational complexity as a consequence of the prediction step. The propagation of the covariance matrix in a prediction step of the KF is divided into two matrix multiplications and a matrix summation, resulting in a computational complexity of $\mathcal{O}(n^3)$ [ValadeEtAl17].

The complexity of a sequential Bayesian update of GMRFs is given by the complexity of solving a sparse linear system of equations. This complexity heavily depends on the sparsity of the information matrix $\mathbf{\Omega}$ which can be expressed by the matrix' bandwidth. This bandwidth is determined by two factors. First, the dimensionality of the GMRF plays an important role, since more edges have to be defined with increasing dimensionality which results in additional non-zero elements in the information matrix. Secondly, with increasing order of the

Table 3.1: An overview of the computational costs of updating the proposed field belief representations where n is the number of field variables and m is the number of taken measurements.

Field Belief	2D	3D
Gaussian Process Regression	$\mathcal{O}(m^3)$	$\mathcal{O}(m^3)$
Kalman Filter	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Kernel DM+V	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Sequential Bayesian GMRFs	$\mathcal{O}\left(n^{\frac{3}{2}}\right)$	$\mathcal{O}(n^2)$

conditional autoregressive process (CAR), the bandwidth of the matrix also increases which results in higher computation time. The computationally most efficient process is the CAR(1) model which is used in this work. According to [RueHeld05], the computational cost of solving a linear system of equations with the sparse information matrix $\mathbf{\Omega}$ is $\mathcal{O}\left(n^{\frac{3}{2}}\right)$ in the two-dimensional case. The computational complexities of the presented belief algorithms are summarized in Tab. 3.1.

3.4.2 Simulated Field Estimation

Depending on the quantity that one wants to explore and map, environmental fields can take many different forms. For example, in the case of hazardous source localization, the field mainly consists of a single peak concentration which decreases with increasing distance from the source. On the other hand, an underwater temperature distribution may be represented by a constant mean with small deviations from that mean function. Both, the former and the latter field examples, have one thing in common: they can be represented by a smooth function. Sometimes, however, an environmental field can also include discontinuities, as in the case of an underwater topographic map which represents the water depth. External objects such as shipwrecks or pipelines evoke jumps in the depth map which are in general complicated to model. Therefore, the goal of this section is to analyze the performance of the belief algorithms for different environmental fields.

In the following simulations, measurements are collected from the environmental fields depicted in Fig. 3.3. All fields are assumed to be constant in time and scaled to a range from zero to ten in x and y, respectively. The fields are explained in more detail below.

Field I The first field, that is to be estimated is a smooth function that is generated by interpolation. Since environmental fields can include sources and

sinks which can be seen as maxima and minima of the field, the field is chosen as depicted in the left of Fig. 3.3.

Field II The second field is an excerpt of the temperature distribution at the surface of the northern sea in the north of Germany. The data is based on satellite observations and has been provided by the Copernicus Marine Service. This field, depicted in the middle of Fig. 3.3, is selected to analyze how the algorithms would work in a real world exploration scenario.

Field III Lastly, a field is investigated which only consists of two field values which are separated by an edge. Since discontinuities can occur in environmental fields, this example is used to check to what extent a jump in the field can be represented.

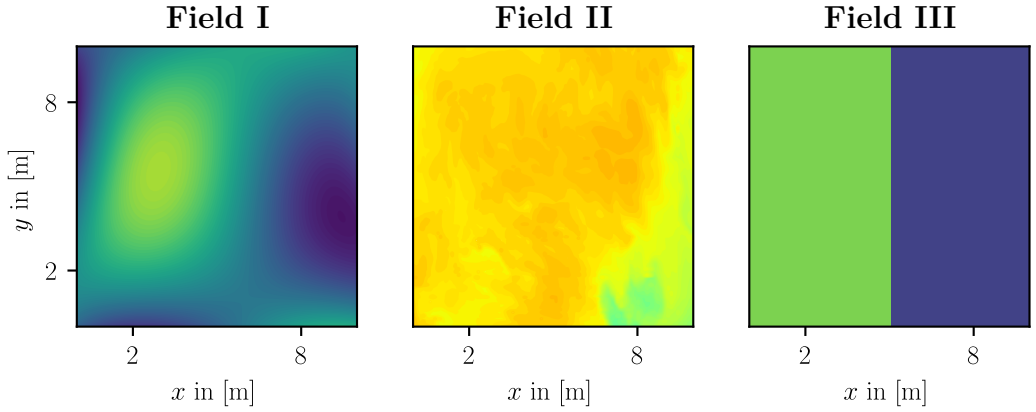


Figure 3.3: Environmental fields that are used for analyzing the performance of the different belief models.

For an analysis of the accuracy of the field representation, the RMSE between the mean belief and the true fields is examined. The beliefs of a single simulation are generated by randomly sampling measurement locations from the true field. A noisy measurement is simulated at the random location and the beliefs are updated. Figure 3.5 shows the mean as well as the 3σ bound of the RMSEs after 30 simulations of each algorithm. The hyper-parameters of the belief models are tuned by hand to achieve good results in terms of the RMSE on the smooth field (**Field I**). One could vary the hyper-parameters for every environmental field but that could be misleading since any true field can include constant field values, smooth functions and even discontinuities. Therefore it is of interest, how well a single field representation with fixed hyper-parameters performs. The parameters of the field belief models are summarized in Tab. 3.2.

Table 3.2: The parameters of the field belief descriptions used for simulations.

Parameter	Symbol	Value
Simulated noise variance	σ_y^2	0.1
GMRF		
Number of grid points	$n_x \times n_y$	30×30
Boundary grid points	$n_{boundary}$	3
Precision value	a	$4 + 10^{-6}$
Kalman Filter		
Number of grid points	$n_x \times n_y$	30×30
Vertical length scale	σ_f	1
Characteristic length scale	ℓ	0.2
Process noise	σ_Q	0.1
Kernel DM+V		
Number of grid cells	$n_x \times n_y$	200×200
Characteristic length scale	ℓ	0.2

The mean beliefs of the presented algorithms after 50 samples have been drawn is examplary shown for **Field I** in Fig. 3.4. It is clearly visible that the mean GMRF belief looks most similar to the actual field. The mean belief of the Kernel DM+V algorithm barely represents the true field since the number of

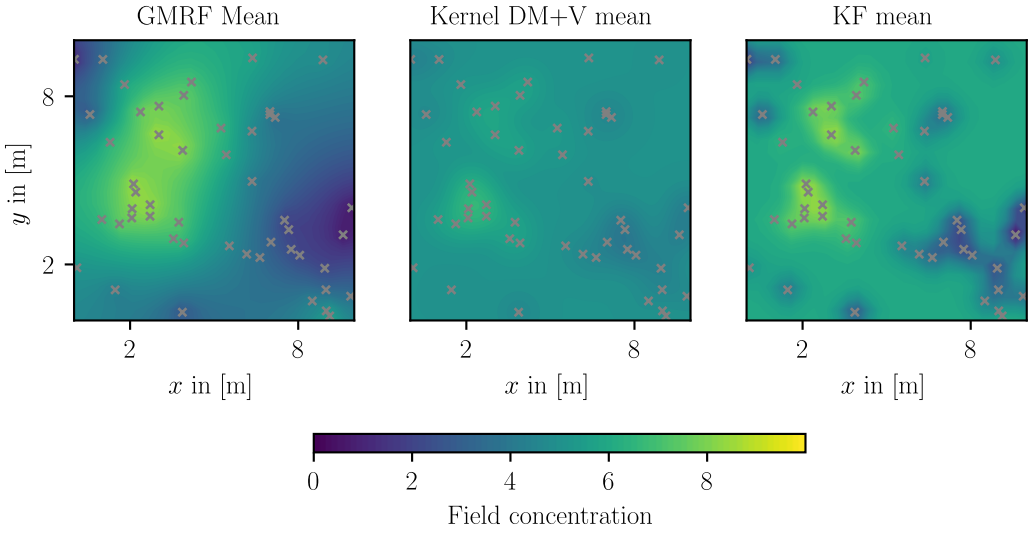


Figure 3.4: Predicted field mean of different belief models after 50 measurements have been simulated. The measurement locations are indicated by grey crosses.

measurements is not sufficient such that the confidence map defined in Eq. (3.43) is close to zero. As a consequence, the predicted field concentrations are generated using the average over all measurements which results in a map with an almost constant field concentration. The mean beliefs of the field representations for **Field II** and **Field III** can be found in Appendix A.3.

As it can be seen in Fig. 3.5, GMRFs have the fastest decaying mean RMSE for all three environmental fields. This is likely because of the latent variable model which models the environmental field as deviations from a constant mean function. This constant mean function is learned simultaneously and is particularly advantageous if only a few samples are available. In the first two cases, where a smooth environmental field is to be mapped, GMRFs have also the highest accuracy in the field representation since the RMSE is always the lowest. The Kalman Filter approach has the lowest rate of convergence but has its strength in representing discontinuities as it outperforms other methods in terms of the RMSE. This could be because the Kalman Filter is roughly speaking a trade-off between the prediction of the process model and the obtained observations. Since the smoothness of the predictions does not match the true field values of **Field III**, the mean belief is shifted to the observations. The Kernel DM+V algorithm has the worst performance in the field accuracy. In contrast to GMRFs and KFs, the Kernel DM+V algorithm is not based on Bayesian Inference but is simply a kernel extrapolation method.

As shown in the left plot of Fig. 3.6, however, the Kernel DM+V is by far the computationally most efficient algorithm. With a mean time of 0.002 seconds for updating the Kernel DM+V field belief, it is 10.5 times faster than the conditioning of GMRFs and 14 times faster than the Kalman Filter. The right plot of Fig. 3.6 shows the mean computation time for a single field update with increasing number of field variables compared to the default number of field variables defined in Tab. 3.2. It is clearly noticeable that the Kalman Filter stands out because of its cubic complexity in the number of field variables. This shows that the scalability of the Kalman Filter approach is limited, as computation times can become intractable for an online application.

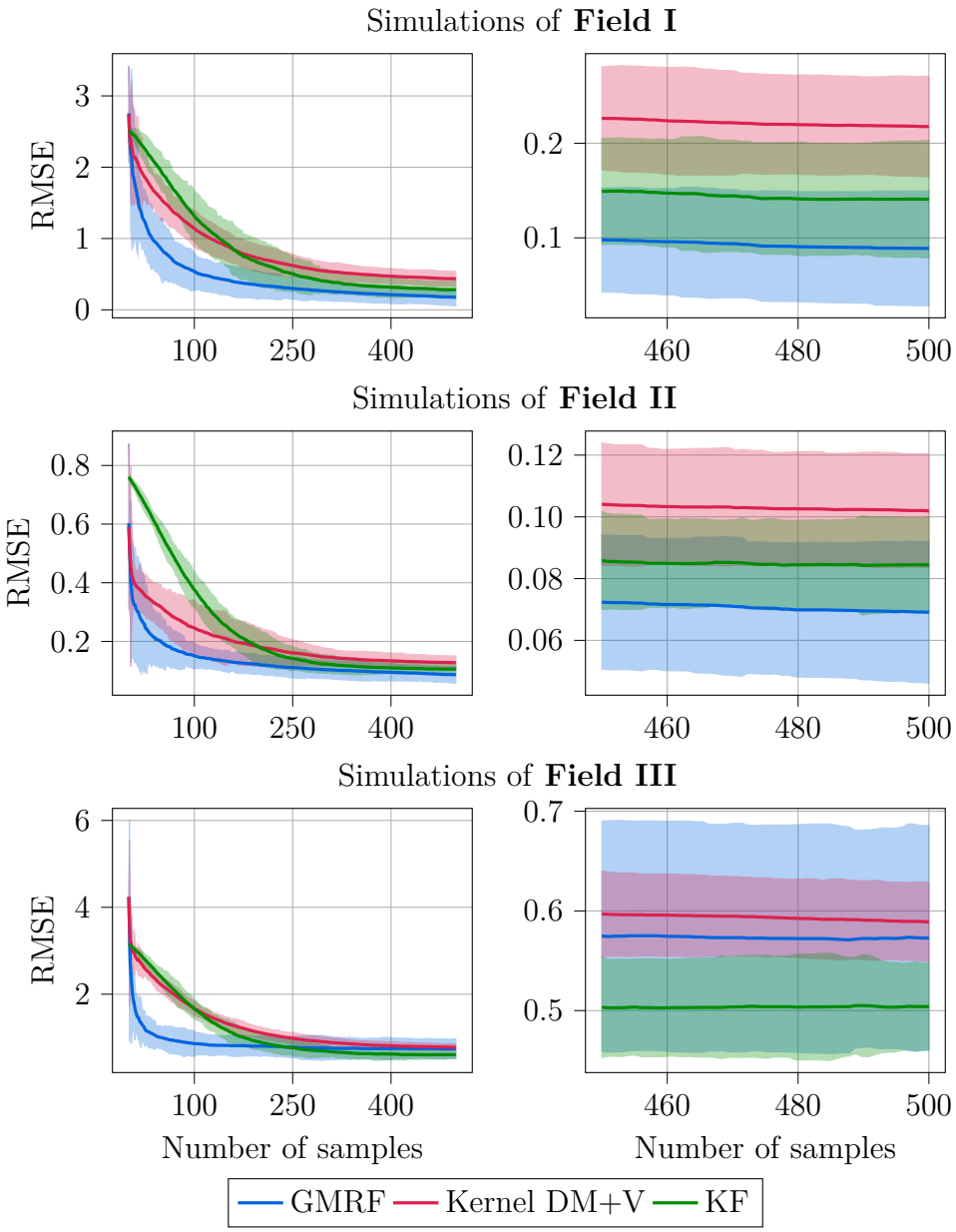


Figure 3.5: Comparison of the RMSE for the different belief models. Fields **I**, **II** and **III** correspond to the first, second and third row, respectively.

3.4.3 Discussion

Table 3.3 provides an overview of the presented algorithms in form of an evaluation matrix. As stated before, the most important criteria for an underwater exploration scenario are chosen to be the accuracy, the scalability and the uncertainty measure of the algorithms. Based on the simulation results, the algorithms

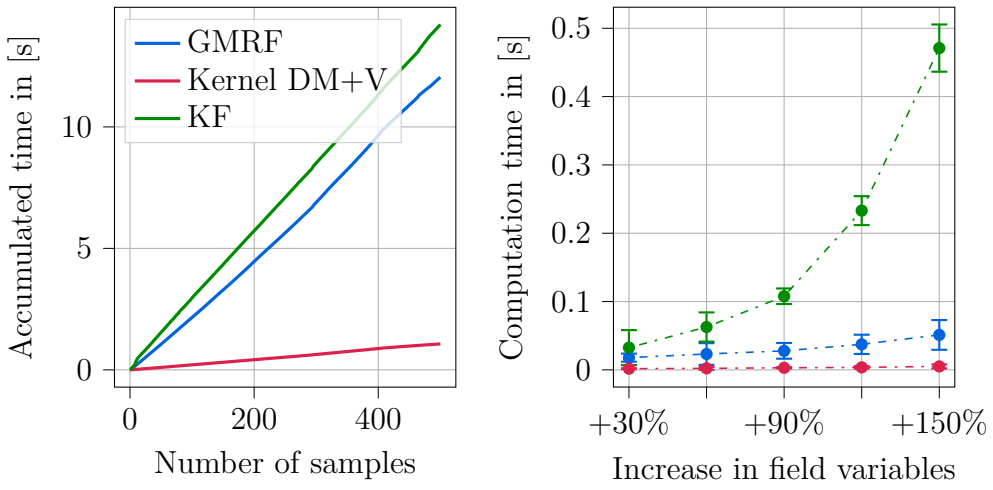


Figure 3.6: Comparison of the computation time. **Left**: Accumulated time needed for updating the field belief in a single simulation. **Right**: Mean and 3σ bound of the computation time for a single belief update with increasing number of field variables.

are rated on a relatively simple scale ranging from bad results (−) to good results (+).

Because of its cubic complexity in the number of field variables, the Kalman Filter has its major drawback in the scalability and hence the computation time. Further, except for the discontinuous field, the accuracy in the field belief is only moderate since densely distributed measurements are needed in order to represent the true field sufficiently well. The uncertainty measure of the KF is given by the covariance matrix of the state. A Bayesian update of the covariance matrix is done after every measurement which also takes into account the sensor's measurement variance. The Kernel DM+V algorithm is superior in the scalability but has a bad accuracy in the field representation. Moreover, the uncertainty map of the modified algorithm is more of a heuristic measure of uncertainty which does not include the measurement variance of the sensor. Therefore, the uncertainty measure is rated as medium (○). Based on the simulation results, GMRFs are the most suitable for an exploration scenario since they provide high accuracy, a sophisticated uncertainty measure as well as a limited computation time. Therefore, GMRFs are used as field representation in this thesis.

Both former works on environmental field modeling and informative path planning, [Geist18] and [Mersch19], suggest that the computational complexity of updating the GMRF belief could be reduced by implementing GMRFs in C/C++ rather than in Python. Although the previous results have been obtained using the 'NumericalPython' (NumPy) library which is already highly optimized, there

Table 3.3: Evaluation matrix for different field belief representations. (+) indicates good performance, (∘) indicates medium performance and (−) stands for bad results.

	Accuracy	Scalability	Uncertainty measure	Overall performance
Kalman Filter	∘	−	+	∘
Kernel DM+V	−	+	∘	∘
GMRFs	+	∘	+	+

is still a remarkable difference in the computation time compared to a C++ implementation, as it can be seen in Fig. 3.7. With a mean time of 0.0017s for a single GMRF update, the C++ implementation is more than eleven times faster than the Python implementation. For that reason, all of the following algorithms are implemented in C++ since the computation time is a crucial factor for the underwater exploration.

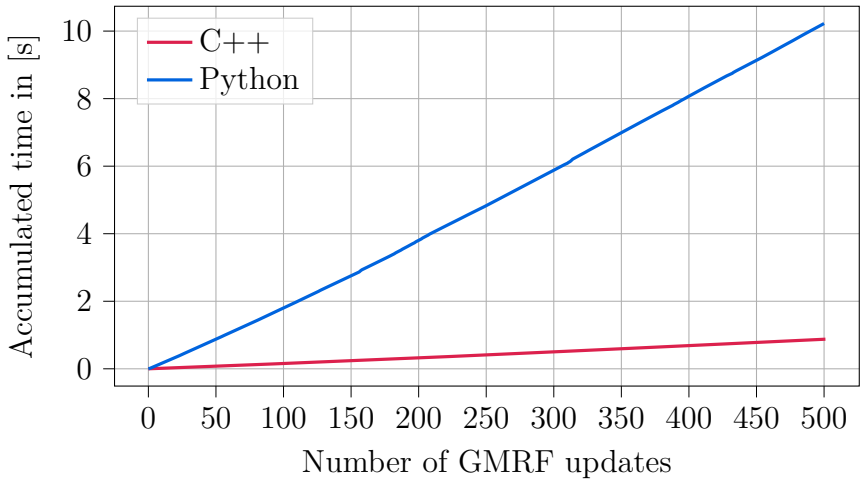


Figure 3.7: Comparison of the accumulated time for updating the GMRF belief in different programming languages.

Chapter 4

Sampling-based Informative Path Planning

This chapter covers the second module of the overall exploration algorithm. The first part on environmental field modeling, presented in the previous chapter, lays the foundation for the informative path planning module which is used to achieve a specific mission goal such as time efficient uncertainty reduction in the field belief.

The task of a path planning module is to generate feasible paths which contribute to the overall mission goal which is often expressed in terms of a cost or reward function. Furthermore, imposed constraints such as actuator saturation or obstacle avoidance complicate the task of finding feasible paths. Once a path is generated, it is passed to a low level control unit of the robot which generates the actuator commands. While following the planned path, new measurements are collected and the field belief is updated sequentially.

Before presenting the path planning algorithm used in this thesis, other state of the art planners are reviewed based on a literature research, in order to find the most suitable for an environmental field exploration.

4.1 Related Work

In the context of informative path planning (IPP), a distinction is often made between non-adaptive and adaptive planning strategies. The former planning methods include among others the coverage planning which is used to explore an environment by application of a sequence of pre-determined actions. Hereby, the goal is to determine a path that passes over all points of an area of interest.

Different strategies of coverage planning are reviewed in [GalceranCarreras13] and include, for example, the typical zigzag path which can be used to explore a spatially unconstrained space. More sophisticated methods deal with the decomposition of the free space into simple, non-overlapping regions which are used to generate a path which passes all points in the free space. These offline path planning methods are typically used in applications where the environment does not change such as for vacuum cleaning robots. Although these methods are computationally efficient, they can not be applied in unknown environments as the pre-determined policy can fail in the case where an obstacle occurs which is not listed in the modeled environment. Therefore, non-adaptive planning strategies are unsuitable for an exploration scenario as a precise map of the environment is often not known in advance. Hence, adaptive planning strategies are of greater interest since these allow to change the path as more information becomes available.

Within the adaptive planning strategies, path planners are further categorized into myopic and non-myopic. The term myopic refers to the shortsightedness of the planner. A classical example for a myopic planner is given in [KrauseSinghGuestrin08], where potential measurement locations are placed greedily with only one step look ahead. As a consequence, a robot which follows a myopic planning strategy is likely to get stuck in local minima. On the other hand, non-myopic planners offer longer planning horizons which can improve the exploration at the cost of higher computational complexity.

In [MaLiuSukhatme16], a non-myopic path planner is utilized for long-term ocean monitoring using AUVs. Hereby, the environment is discretized into a grid map with hierarchical structure. On the top layer, a grid map of low resolution is defined where each grid cell is again discretized into a grid map of higher resolution. The non-myopic planner is then used to generate paths on each hierarchical grid map by formulation of a *Markov Decision Process* (MDP) which is solved using dynamic programming. To be more precise, the outcome of this planner is a sequence of grid cells which have to be traversed by the robot. The hierarchy enables scalability to areas of arbitrary size since another level of hierarchy can always be added. However, a discretization of this form does not allow for continuous path planning as the measurement locations are determined by a fixed set of grid points.

Another non-myopic approach which offers path planning in continuous space is presented in [MorereMarchantRamos17] and extended for an application to underwater field exploration in [Mersch19]. The IPP problem is formulated as a partially observable Markov decision process (POMDP) which is a probabilistic framework for sequential decision making where a sequence of actions is determined that maximizes a reward function. In order to obtain an approximate solution of the POMDP, Monte Carlo tree search (MCTS) is utilized which

has proven to effectively plan on large POMDPs, [MorereMarchantRamos17]. Roughly speaking, MCTS balances the rate of exploration and exploitation of the search tree in the sense that either existing branches are deepened or new branches are explored. By formulating the reward function as a Gaussian process, Bayesian optimization (BO) is used to allow for continuous action selection resulting in continuous trajectories of sampling locations. Although, the proposed algorithm can find close-to-optimal solutions in continuous space, its major drawback is the scalability as the computational complexity significantly depends on the depth of the search tree and hence the length of the planned paths.

To overcome the problem of computational limitations, prominent tools are the sampling based planning strategies such as the *Rapidly-exploring Random Trees* (RRTs), first presented in [Lavalle98]. RRTs are widely used in the context of motion planning where a feasible trajectory with predefined start and end point is to be planned. A modification which has received a lot of attention in recent years is the RRT* algorithm which, in contrast to RRTs, is said to be asymptotically optimal with increasing computation time [KaramanFrazzoli11]. In [HollingerSukhatme14], the RRT* algorithm is used to solve the IPP problem with an imposed budget constraint yielding the *Rapidly-exploring Information Gathering* (RIG) trees. This is done in a non-adaptive fashion where the tree is built once to obtain an offline generated path which can be traversed by a robot. An adaptive implementation of RIG trees for informative planning can be found in [ViserasShutinMerino19] where the entire tree is rebuilt from scratch whenever a new path is planned. This method can be applied to online field exploration but it has two main disadvantages. First, parts of the tree are discarded in every iteration as only the best branch of the tree is stored and executed. In the case of IPP this can be computationally demanding since every node of the tree is assigned with an information metric which has to be calculated while growing the tree. If the tree is rebuilt every time a new path is obtained, the information metric is calculated repeatedly which is unnecessary and also unsuitable for computationally constrained platforms. Second, the RRTs are often generated with a limited planning horizon which can result in sub-optimal paths. A recently proposed improved algorithm is a real-time variant of RRT* called RT-RRT* which continuously expands and maintains a single tree of possible trajectories. This algorithm, presented in [NaderiRajamakiHamalainen15], is used for planning paths from an initial position to a goal position in 2D space while avoiding static and dynamic obstacles. The maintenance of a single tree that is spread in the entire space of interest solves the two aforementioned problems of other online RRT* implementations, namely the repeated rebuilding of the tree and the limited planning horizon. Therefore, the main focus of this thesis is on RT-RRT*.

The goal of this chapter is to extend the concepts of RT-RRT* such that it can be used as the path planning module for an environmental field exploration, yielding

the Information Gathering RT-RRT* (IG-RT-RRT*) algorithm. This approach combines the advantages of both algorithms, the RIG trees and RT-RRT*. The resulting planner is able to generate informative paths with an unlimited planning horizon in the area of interest and a sophisticated obstacle avoidance which can deal with suddenly occurring obstacles.

4.2 Information Metrics

Before presenting the informative path planning algorithm, the notion of uncertainty has to be specified first. A formulation of the uncertainty about the unknown field distribution is needed in order to plan paths that allow for an exploration of the entire field. This section reviews two metrics that are commonly used to evaluate measurement locations regarding their uncertainty. This is essential for the path planning module to obtain the most informative path of potential measurement locations.

According to [KrauseSinghGuestrin08], a natural notion of uncertainty in a GP framework is the *conditional entropy* $\mathcal{H} \in \mathbb{R}$ of the set of random variables $\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}$, given previous observations $\boldsymbol{f}_{\mathcal{A}}$. Here, $\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}$ refers to the unobserved random variables at locations $\mathcal{X}_{\mathcal{F}\setminus\mathcal{A}}$ and $\boldsymbol{f}_{\mathcal{A}}$ is the set of measurements at observed locations $\mathcal{X}_{\mathcal{A}}$. The entire space is denoted by $\mathcal{F}$ and $\mathcal{A}$ is a subset of locations that have been visited by the robot. The scalar conditional entropy can be obtained as

$$\mathcal{H}\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}|\boldsymbol{f}_{\mathcal{A}}\right) = -\int_{\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}}\int_{\boldsymbol{f}_{\mathcal{A}}} p\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}, \boldsymbol{f}_{\mathcal{A}}\right)\log\left(p\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}|\boldsymbol{f}_{\mathcal{A}}\right)\right)\mathrm{d}\boldsymbol{f}_{\mathcal{A}}\mathrm{d}\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}} \quad (4.1)$$

and provides a measure of uncertainty for the unobserved locations given the observed ones. In this sense, Eq. (4.1) describes the overall uncertainty of the field. Accordingly, the conditional entropy of a single random variable f_y can be obtained by substitution of f_y for $\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}$ which can be calculated in closed form for Gaussian densities, yielding

$$\begin{aligned}\mathcal{H}\left(f_y|\boldsymbol{f}_{\mathcal{A}}\right) &= \frac{1}{2}\log\left(2\pi\sigma^2_{f_y|\boldsymbol{f}_{\mathcal{A}}}\right) && (4.2)\\ &= \frac{1}{2}\log\left(\sigma^2_{f_y|\boldsymbol{f}_{\mathcal{A}}}\right) + \frac{1}{2}\left(\log\left(2\pi\right)+1\right). && (4.3)\end{aligned}$$

Hence, the conditional entropy of a single random variable given previous measurements is proportional to the logarithm of its conditional variance $\sigma^2_{f_y|\boldsymbol{f}_{\mathcal{A}}}$.

If a set of potential measurement locations $\boldsymbol{f}_y$ is given, the most informative measurement location with respect to the conditional entropy can be determined

by maximizing $\mathcal{H}$, reading

$$f_{y_i}^* = \underset{f_{y_i} \in \boldsymbol{f}_y}{\arg\max}\, \mathcal{H}\left(f_{y_i} | \boldsymbol{f}_{\mathcal{A}}\right). \tag{4.4}$$

However, as stated in [KrauseSinghGuestrin08], measurement locations that are obtained using an entropy criterion are often placed along the boundary of the space of interest as the predictive variance σ^2 is large in these areas. Since measurements enable predictions of environmental field values in a neighborhood of the measurement locations, this is counterproductive as part of the information is lost at the boundary. Consequently, a measurement location that is placed in the center of the field achieves the highest decrease in the fields entropy.

An improved uncertainty measure is given by the *mutual information* $\mathcal{I}$ between unobserved field values $\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}$ and measurements $\boldsymbol{f}_{\mathcal{A}}$. In contrast to the entropy criterion, mutual information describes the entropy decrease in the set of unobserved locations after taking measurements $\boldsymbol{f}_{\mathcal{A}}$. This decrease can be expressed as

$$\mathcal{I}\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}, \boldsymbol{f}_{\mathcal{A}}\right) = \mathcal{H}\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}\right) - \mathcal{H}\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}} | \boldsymbol{f}_{\mathcal{A}}\right) \tag{4.5}$$

where the entropy of the unobserved values before taking the measurements $\mathcal{H}\left(\boldsymbol{f}_{\mathcal{F}\setminus\mathcal{A}}\right)$ can be obtained as in Eq. (4.3) with the prior variance substituted for the conditional variance. Using the mutual information criterion has the effect, that measurement locations are more likely to be placed in the center of the field as this achieves the highest entropy decrease.

An example for optimal placement of measurement locations is given in Fig. 4.1. To simplify the problem, the space is discretized into a finite set of measurement locations, indicated by grey crosses. The exploring robot has its initial position at $(x, y) = (2, 2)$m with a heading in positive y direction. The heading is used to constrain the motion of the robot such that movement which is perpendicular to the robot's heading is not possible. Then, a path of measurement locations for different information metrics is generated using dynamic programming. In this way, the optimal path with respect to the information metric can be obtained for a fixed planning horizon. The left plot in Fig. 4.1 shows the planned path using the entropy criterion which results in a path that has its measurement locations placed along the border of the space. In contrast, a planned path based on the MI criterion avoids the boundaries, except for the last measurement location, as it can be seen on the right.

Fig. 4.1 also provides an unconventional interpretation of the information metrics. When using the entropy criterion, the goal is to "collect" as much bright colors as possible along the path, where the bright colors represent areas of high uncertainty. The MI criterion, on the other hand, has the effect that measurements are

placed such that the color of the whole field is as dark as possible. Although MI provides an improved metric for the information of measurement locations, its calculation requires more computational effort than the entropy criterion. The entropy can be obtained by simply evaluating the predictive variance at a location which is known a priori. When selecting a measurement location for the MI criterion, the field belief has to be updated with a simulated measurement in order to evaluate the posterior entropy. As shown in the previous chapter, a field belief update can be computationally expensive which is unsuitable for sampling-based planning methods as many potential measurement locations need to be evaluated regarding their uncertainty. Therefore, the entropy is used as a measure of uncertainty in this thesis.

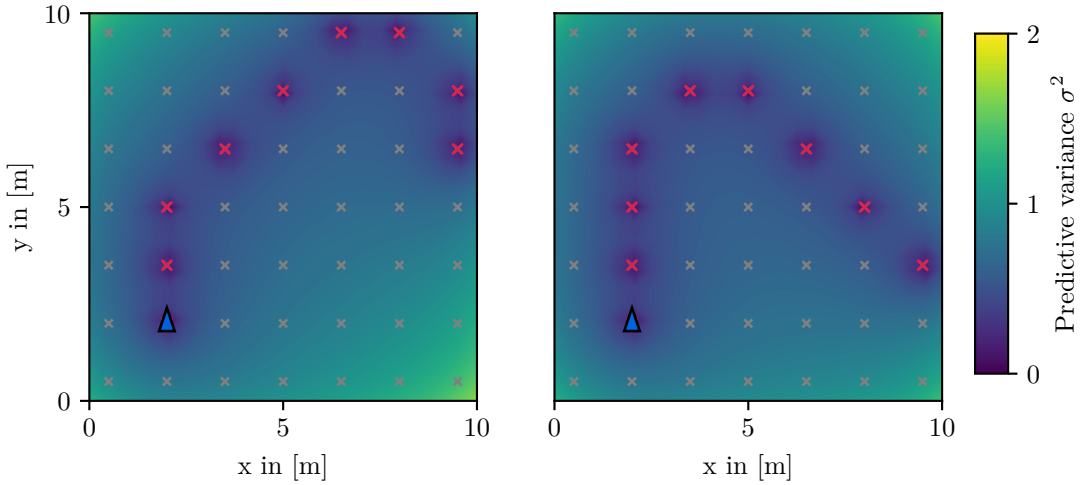


Figure 4.1: Exemplary paths that are planned in a grid world using different information metrics. The set of potential measurement locations is indicated by grey crosses. The planned path is colored in red. **Left**: Measurement locations that maximize the entropy along the path. **Right**: A path that is generated using the MI criterion.

4.3 Information Gathering RT-RRT*

In this section, the Information Gathering Real-Time RRT* (IG-RT-RRT*) algorithm is proposed which can be used to generate paths for an autonomously exploring robot. This path planning algorithm is an extension of the original RT-RRT* algorithm which is mainly used for planning paths with predefined start and end points. In the original RT-RRT* algorithm, the entire tree is kept alive and is continually optimized while the robot is moving through the environment. In the proposed extension, each tree node is assigned with an information metric

which can be efficiently updated with the knowledge of the GMRF belief. This enables the planning of informative paths in an anytime fashion, meaning that an informative path is quickly identified and improved with increasing computation time. Moreover, by maintenance of a single tree of possible trajectories in the entire space of interest, long planning horizons are possible which enable non-myopic planning behavior. The sophisticated obstacle avoidance of the RT-RRT* algorithm further allows to explore spatially constrained environments with suddenly occurring obstacles since fast replanning is possible.

In [NaderiRajamakiHamalainen15], the RT-RRT* algorithm is used to generate trajectories which minimize the Euclidean distance along the path, resulting in a tree of possible trajectories connecting the root of the tree to any point in space with minimum distance. In this work, the distance plays only a minor role, as the actual goal is to minimize the uncertainty in the field belief. Therefore, rather than using a Euclidean distance cost function to evaluate the quality of a path, paths are evaluated using the notion of utility. This utility function defines the trade-off between informativeness and costs of the paths, where costs can be defined as the traveled distance or the actuator commands.

An overview of the interactions of the individual modules that are needed for an autonomous exploration is depicted in Fig. 4.2. In order to plan an informative path $\mathcal{P}$ for an environmental field exploration, knowledge about the state of the robot as well as the state of the environment is needed. The robot's state is given by a state estimation module which provides knowledge about the position and the orientation of the robot. The estimation of the robot's state is particularly challenging in the underwater domain and goes beyond the scope of this work. Therefore, it is assumed that the the state of the robot is known by the planning module. The reader is referred to [WatsonDueckerGroves20], for an insight into possible solutions to the underwater localization problem. For the IG-RT-RRT* algorithm, a 2D pose of the robot is sufficient for planning. This pose consists of a position (x, y) in an inertial frame as well as an orientation φ, as depicted in Fig. 4.2. The state of the environment is given by the GMRF belief which is continually updated during the exploration, see Section 3.2. As the tree is sufficiently large, the most informative path $\mathcal{P}$ from the current position of the robot to a goal position can be extracted and tracked by a control unit. While the robot is following a path, the planning tree is changed in a way that at any point in time, another path can be chosen which allows for fast replanning.

The main loop of the IG-RT-RRT* algorithm is illustrated in Fig. 4.3. The heart of the planning algorithm is the tree $\mathcal{T}$ which provides a set of possible paths that can be traversed by the robot. The tree is expanded until it is sufficiently large meaning that enough paths are available to explore an environmental field. Thus, the expansion increases the number of possible paths in the tree. The quality of the paths is improved by optimizing the edges of the planning tree through the

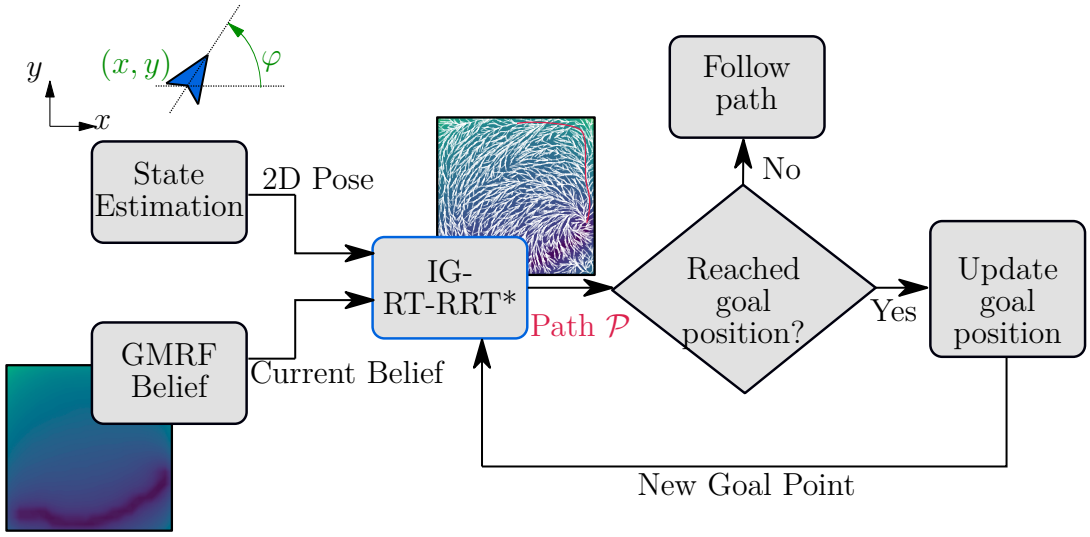


Figure 4.2: An overview of the path planning module embedded into the exploration framework.

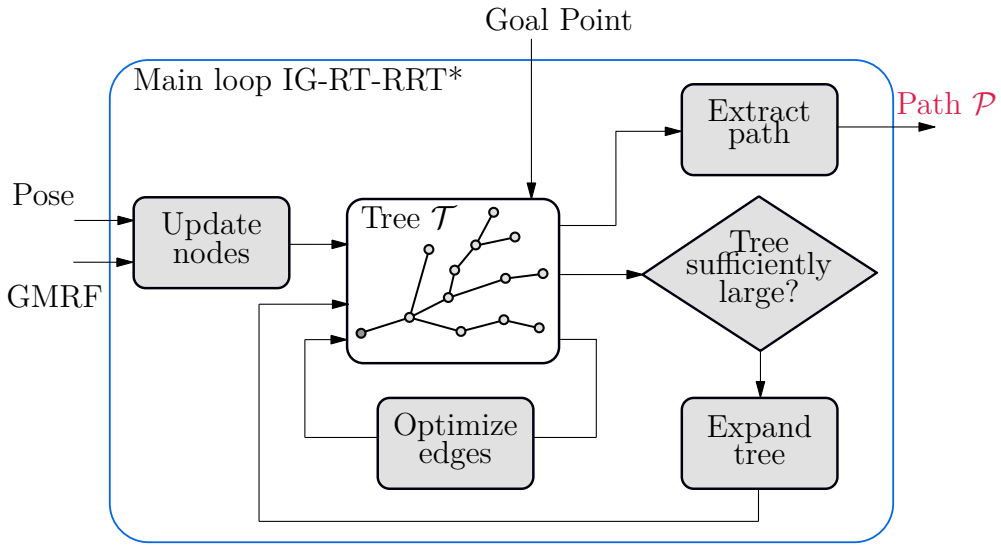


Figure 4.3: A zoom into the Information Gathering RT-RRT* algorithm.

so called rewire mechanism. The nodes of the tree are updated, whenever the robot's pose or the GMRF belief are updated. Lastly, the path $\mathcal{P}$ is extracted from the tree.

The remainder of this chapter is structured as follows. First, the notation and the problem setup are covered in Section 4.3.1. The expansion of the tree and the optimization of the edges through rewiring is explained in Sections 4.3.2 and

4.3.3, respectively. The update of nodes due to a new pose estimate is described in Section 4.3.4. The extraction of a path from the tree is covered in Section 4.3.5. In order to also explore spatially constrained environments, an obstacle avoidance is needed which is presented in Section 4.3.6. To improve the performance of the proposed algorithm in terms of computational storage and computation time, some implementation details are reviewed in Section 4.3.7. Finally, the cost, the information and the utility along a path are formulated in Section 4.3.8 which enable the planning of informative paths. This structure is chosen in order to first present the construction and maintenance of the planning tree which is then utilized to better illustrate the individual components of the utility function.

4.3.1 Notation and Problem Setup

This section briefly covers the notation which is used in the derivation of the planning algorithm. Further, the problem of finding an informative path is formulated as a maximization problem which can then be solved with the proposed planning algorithm.

Let the work space, where the robot is operating be denoted by $\mathcal{X}$. This workspace is a subset of $\mathbb{R}^2$, since this work aims to explore environmental fields in two dimensions. Further, $\mathcal{X}$ is bounded such that the robot can not exceed the area of interest. Within $\mathcal{X}$, the workspace can be occupied by obstacles given by $\mathcal{X}_{obs} \subsetneq \mathcal{X}$ which can be extended as new obstacles occur. The free space, where the robot is allowed to move, is denoted by $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$. The planning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is built in the free space and consists of vertices (or nodes) $\mathcal{V}$ and connecting edges $\mathcal{E}$. If two nodes V_i and V_j are connected by an edge, hence $\{i, j\} \in \mathcal{E}$, the nodes are considered to be on a path segment. Each node

$$V_i = \{\boldsymbol{p}_i, c_i, g_i, U_i\} \tag{4.6}$$

consists of a pose $\boldsymbol{p}_i = [x_i, y_i, \varphi_i]^T$, a cost c_i, an information g_i and the associated utility U_i. This decomposition in individual quantities allows to efficiently update the entire tree, as, for example, only the information gain g_i depends on the GMRF belief. The pose $\boldsymbol{p}_i$ is defined by the position $\boldsymbol{x}_i$ and the orientation φ_i, as illustrated in Fig. 4.2. In the following, the shorthand notations $c(V_i)$ or $g(V_i)$ are used to refer to the quantities of a specific node. The root node of the tree is denoted by V_0 and is located at the current position of the robot. The root node is considered to be the parent node of all nodes V_i which are connected to the root node. Accordingly, the connected nodes are said to be child nodes of V_0.

The problem of finding an informative path can be stated as follows. The goal is to find a path $\mathcal{P}^* = (\boldsymbol{x}_0^*, \boldsymbol{x}_1^*, \dots, \boldsymbol{x}_k^*)$ which consists of measurement locations $\boldsymbol{x}_i^*$ which should be traversed by the robot. This path should maximize the utility

along the path which can be stated as

$$\mathcal{P}^* = \arg\max_{\mathcal{P}\in\mathcal{T}} U(\mathcal{P}). \tag{4.7}$$

Additional constraints such as obstacle avoidance or actuator saturation can be imposed by including penalty terms in the utility function. One can think of the path selection in the following way. A tree $\mathcal{T}$ is generated which provides a set of possible paths to any point in space. All paths from the root node V_0 to an arbitrary node V_i are generated such that the utility is maximized along that path. If the number of nodes is approaching infinity, all paths starting at the root node are optimal with respect to the utility function [KaramanFrazzoli11]. However, since the number of nodes is always finite, optimality can not be guaranteed. Then, within the set of possible paths $\mathcal{T}$, the path with maximum utility is selected, according to Eq. (4.7). Since the formulation of the utility function is covered at the end of this chapter, it might help to think of the utility function as negative distance in the following. This would mean that maximizing utility corresponds to minimizing the distance of the paths.

The algorithmic structure of the proposed Information Gathering RT-RRT* algorithm is shown in Algorithm 1. The main loop consists of five sub algorithms that are executed to obtain an informative path. First, the tree needs to be expanded, meaning that new nodes are added to the tree (line 5). The edges between nodes are changed in the Rewire() algorithm which optimizes the paths with respect to the utility function. These algorithms are executed for a limited user-defined computation time which enables the real-time applicability. Whenever new measurements are available, the GMRF belief is updated which also updates the information g_i of the tree nodes (line 8). If the robot follows a path, it changes its pose and thus, the position and orientation of the root node have to be changed accordingly (line 11). Lastly, the most informative path is extracted from the tree. The individual algorithms are presented in the following.

4.3.2 Tree Expansion

In order to generate paths in large parts of the free space, the tree $\mathcal{T}$ needs to be expanded first. This expansion algorithm is summarized in Algorithm 2 and illustrated in Fig. 4.4. The expansion is done incrementally, where a point $\boldsymbol{x}_{rand}$ is randomly sampled from the free space $\mathcal{X}_{free}$. Then, the closest node in the tree is obtained as potential parent node using a distance function, e.g. the Euclidean distance. The function steer $(\cdot,\cdot)$ is introduced, as the sampled point can be far away from the actual tree. This function takes two input arguments $\boldsymbol{x}_{closest}$ and $\boldsymbol{x}_{rand}$ and returns a point $\boldsymbol{x}_{new}$ which is located on the connecting line between $\boldsymbol{x}_{closest}$ and $\boldsymbol{x}_{rand}$ with a desired distance, see Fig. 4.4. This has the effect that

Algorithm 1 Information Gathering RT-RRT*

1: **Inputs**: $\boldsymbol{x}_0$
2: Initialize $\mathcal{T}$ with root node: $\mathcal{V} \leftarrow \{V_0\}, \mathcal{E} \leftarrow \emptyset$
3: **loop**
4: **while** time left for Expansion and Rewiring **do**
5: Expand() and Rewire()
6: **end while**
7: **if** New measurement available **then**
8: UpdateInformation()
9: **end if**
10: **if** Robot moved **then**
11: UpdateRootNode()
12: **end if**
13: $\mathcal{P} \leftarrow$ ExtractPath()
14: **end loop**

all edges have roughly the same length. Next, a new node V_{new} at the position $\boldsymbol{x}_{new}$ with its potential parent $V_{closest}$ is created but not yet added to the tree. The Node $(\cdot)$ operator (line 6) initializes all quantities of a node as introduced in

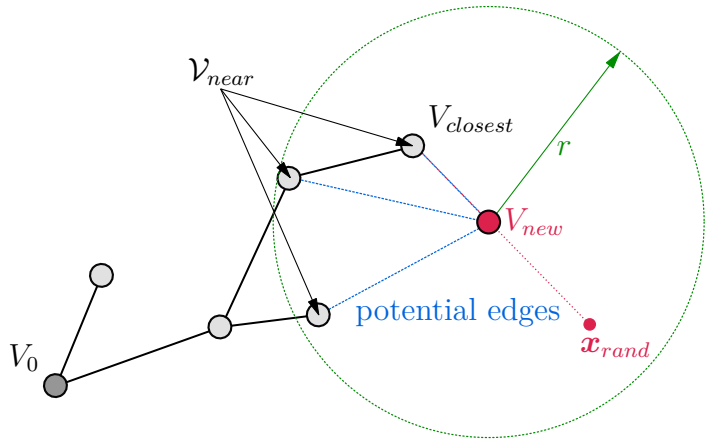


Figure 4.4: Illustration of the tree expansion where $\boldsymbol{x}_{rand}$ is the sampled point and the near nodes within the circle of radius r are given by the set $\mathcal{V}_{near}$.

Algorithm 2 Expand()

1: **Inputs**: $\mathcal{T}, \mathcal{Q}_r, \mathcal{Q}_s, r, \mathcal{V}_{max}$
2: **if** $|\mathcal{V}| \leq \mathcal{V}_{max}$ **then**
3: $\quad \boldsymbol{x}_{rand} \leftarrow \text{sample}(\mathcal{X}_{free})$
4: $\quad V_{closest} \leftarrow \arg\min_{V_i \in \mathcal{V}} \text{dist}\left(\boldsymbol{x}_{rand}, \boldsymbol{x}\left(V_i\right)\right)$
5: $\quad \boldsymbol{x}_{new} = \text{steer}\left(\boldsymbol{x}\left(V_{closest}\right), \boldsymbol{x}_{rand}\right)$
6: $\quad V_{new} = \text{Node}\left(\boldsymbol{x}_{new}, V_{closest}\right)$
7: $\quad$ **if** $\text{line}\left(\boldsymbol{x}_{new}, \boldsymbol{x}\left(V_{closest}\right)\right) \in \mathcal{X}_{free}$ **then**
8: $\quad\quad \mathcal{V}_{near} \leftarrow \text{NearNodes}\left(V_{new}, \mathcal{V}, r\right)$
9: $\quad\quad U_{max} = U\left(V_{new}\right)$
10: $\quad\quad$ **for all** $V_i \in \mathcal{V}_{near}$ **do**
11: $\quad\quad\quad V = \text{Node}\left(\boldsymbol{x}_{new}, \boldsymbol{x}\left(V_i\right)\right)$
12: $\quad\quad\quad$ **if** $\text{line}\left(\boldsymbol{x}\left(V_{new}\right), \boldsymbol{x}\left(V_i\right)\right) \in \mathcal{X}_{free}$ and $U\left(V\right) > U_{max}$ **then**
13: $\quad\quad\quad\quad V_{new} = V, \;\; U_{max} = U\left(V\right)$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{V_{new}\}, \;\; \mathcal{E} \leftarrow \mathcal{E} \cup \{(V_{new}, parent\left(V_{new}\right))\}$
17: $\quad\quad$ push V_{new} to the front of $\mathcal{Q}_r$
18: $\quad$ **end if**
19: **end if**

Eq. (4.6) and is defined as

$$\text{Node}\left(\boldsymbol{x}_c, V_p\right) := \Bigg(\boldsymbol{p}_c = \left[\boldsymbol{x}_c, \tan^{-1}\left(\frac{x_c - x\left(V_p\right)}{y_c - y\left(V_p\right)} \right) \right]^T, \;\; c_c = c\left(V_p\right) + c_{c \to p}, \\ g_c = g\left(V_p\right) + g_{c \to p}, \;\; U_c = U\left(c_c, g_c\right) \Bigg) \tag{4.8}$$

which takes a position $\boldsymbol{x}_c$ and a parent node V_p as input arguments. The heading φ_c of the new node is given by the angle of the connecting edge. The cost c_c along the path can be calculated by summation of the parents cost $c\left(V_p\right)$ and the cost between nodes which is given by the subscript $c \to p$. The same holds for the information g_c. The utility U_c of a node is a function of costs and information which cannot be calculated in recursive form.

In the case of classic RRTs, the expansion would be completed at this point. In order to improve the paths, however, the nodes in a neighborhood $\mathcal{V}_{near}$ of the newly created node are reviewed as potential parents (line 8-15). All nodes within a rewiring radius r around $\boldsymbol{x}\left(V_{new}\right)$ are taken into consideration and the edge of V_{new} to its parent is chosen such that the utility along the path is maximized, as it can be seen in Fig. 4.4. Having found the best possible parent node, V_{new} is added to the tree (line 16).

Repeating this process over and over again until a maximum number of nodes $\mathcal{V}_{max}$ is reached results in a large tree that is spread in the entire free space $\mathcal{X}_{free}$. Further, it is ensured that the paths are collision free since whenever a potential edge is created, it is checked whether the connecting line collides with an obstacle (line 7 of Algorithm 2). Although a new node V_{new} is always connected to the best possible parent, it is not checked whether V_{new} is a potentially better parent for nodes in the neighborhood. This local optimization of edges is done in the rewiring step which is presented in the following.

4.3.3 Tree Rewiring

The rewire mechanism is the essential sub algorithm of RRT* which enables asymptotic optimality. Hereby, for an arbitrary node of the tree, it is checked whether this node is a better parent for nodes in a neighborhood. In the RT-RRT* algorithm it is distinguished between two different rewire mechanisms. First, the rewiring of random nodes and, second, the rewiring around the root node. Both mechanisms are used to work through a priority queue given by $\mathcal{Q}_r$ for the random nodes and $\mathcal{Q}_s$ for the nodes around the root of $\mathcal{T}$, respectively. As stated in line 17 of Algorithm 2, each new node that is added to the tree is automatically pushed to $\mathcal{Q}_r$ and therefore processed in the rewiring step of random nodes. This procedure is shown in Algorithm 3. Given the queue $\mathcal{Q}_r$, the node with the highest priority V_r is extracted for rewiring (line 3). Then, for all nodes V_i which are within a radius r around V_r, it is examined whether the utility increases if V_r is chosen to be the new parent of V_i. If this is the case, the edges of $\mathcal{T}$ are updated and V_i is pushed to the back of $\mathcal{Q}_r$, meaning that it is checked if V_i is a better parent for neighboring nodes. Whenever a parent of a

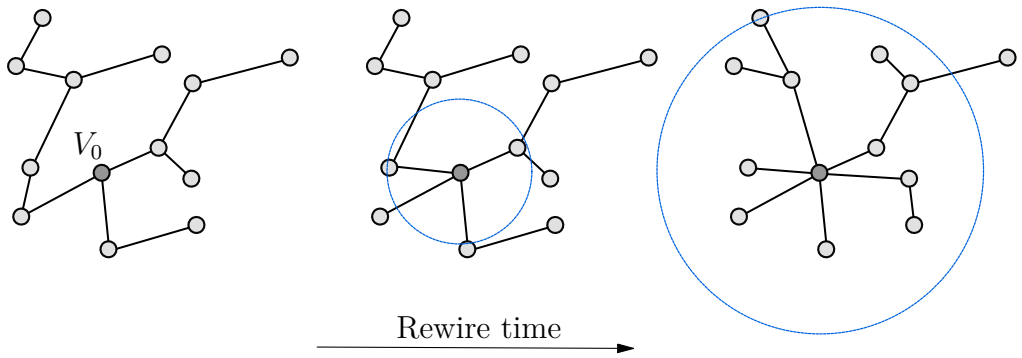


Figure 4.5: Illustration of the rewire from root mechanism which is used to optimize the edges of the tree with increasing rewire time. In this example, the Euclidean distance along a path is minimized.

Algorithm 3 Rewire Random Nodes

1: **Inputs**: $\mathcal{T}, \mathcal{Q}_r, r$
2: **repeat**
3: $V_r \leftarrow \text{PopFirst}(\mathcal{Q}_r)$
4: $\mathcal{V}_{near} \leftarrow \text{NearNodes}(V_r, \mathcal{V}, r)$
5: **for all** $V_i \in \mathcal{V}_{near}$ **do**
6: $c_{old} = c(V_i), g_{old} = g(V_i), \quad c_{new} = c(V_r) + c_{i \to r}, g_{new} = g(V_i) + g_{i \to r}$
7: $U_{old} = U(V_i), \quad U_{new} = U(c_{new}, g_{new})$
8: **if** $\text{line}(\boldsymbol{x}(V_i), \boldsymbol{x}(V_r)) \in \mathcal{X}_{free}$ and $U_{new} > U_{old}$ **then**
9: $\mathcal{E} \leftarrow (\mathcal{E} \setminus \{(V_i, parent(V_i))\}) \cup \{(V_r, V_i)\}$
10: PropagateValues(V_i)
11: Push V_i to the end of $\mathcal{Q}_r$
12: **end if**
13: **end for**
14: **until** time is up or $\mathcal{Q}_r$ is empty

node V_i is changed, it is important to change the values of cost, information and utility for all nodes in the sub tree of V_i, since the cost c and the information g are recursively calculated using the parent node. This propagation of values can therefore be done in a recursive fashion starting with the update of values for V_i. Next, the values of all child nodes of V_i are updated and propagated to their child nodes. This is repeated until the values of the leaf nodes are updated. Leaf nodes are characterized by the fact that they do not have any child nodes. This propagation of values is summarized in line 10 of Algorithm 3. As new nodes are added to $\mathcal{Q}_r$ in line 11, the rewiring step can take some computation time, especially if the tree is large. For that reason, the time for rewiring of random nodes is limited by a user-defined duration.

The second rewire mechanism, which is important for replanning while the robot moves, is the rewiring from root, summarized in Algorithm 4. This sub system is very similar to the random rewiring with the main difference that the focus point is on the root of the tree and not on a random node in the tree. The priority queue $\mathcal{Q}_s$ always starts with the root node V_0. Then, in the same way as described in Algorithm 3, the root node is set as potential parent for all neighboring nodes and if the utility increases, the edges of the tree are updated. In contrast to the random rewiring, all of the near nodes are pushed to $\mathcal{Q}_s$ if they have not yet been checked for rewiring. One can picture this rewiring procedure as locally optimizing paths within a circle which is centered at the root node. This circle starts with a zero radius and grows with increasing rewire time. If enough time is available, the entire tree gets rewired, resulting in the best possible paths with respect to the utility function given the number of nodes. On the other hand, if the rewiring time is limited, the paths starting at the root node are

Algorithm 4 Rewire From Root

1: **Inputs**: $\mathcal{T}, \mathcal{Q}_s, r$
2: **if** $\mathcal{Q}_s$ is empty **then**
3: Push V_0 to $\mathcal{Q}_s$
4: **end if**
5: **repeat**
6: $V_r \leftarrow \text{PopFirst}(\mathcal{Q}_s)$
7: execute lines 4 - 13 without line 11 of Algorithm 3
8: **if** $V_i \notin \mathcal{Q}_s$ and V_i has not been rewired **then**
9: Push V_i to the end of $\mathcal{Q}_s$
10: **end if**
11: **if** Root Node V_0 is moved **then**
12: $\mathcal{Q}_s \leftarrow \emptyset$
13: **end if**
14: **until** time is up or $\mathcal{Q}_s$ is empty

locally optimized within some radius. An illustrating example, of a tree $\mathcal{T}$ that is rewired starting from the root is shown in Fig. 4.5. In this case, the goal is to minimize the Euclidean distance of the paths, hence the length of the connecting edges. As the rewire time increases, more nodes are added to $\mathcal{Q}_s$ where the nodes that are further away are less prioritized. The resulting optimized tree is depicted in the right of Fig. 4.5. Note that the optimal solution would be to connect all nodes directly to the tree root which is, however, not possible as the maximum distance between nodes is given by the user-defined rewire radius r.

A consequence of the rewiring step is that connecting edges of the tree do not necessarily have equal length as all nodes within the radius r can be connected. This has to be taken into account when designing the cost and information between nodes.

4.3.4 Pose Update

So far the expansion and rewiring of the tree have been presented. These two steps are essential for generating a set of possible paths that can be followed by the robot. However, the root node V_0 was assumed to be static with a fixed pose $\boldsymbol{p}_0 = [x_0.y_0, \varphi_0]^T$ which is not the case as the robot is moving through the environment. If a new pose estimate is available, the root node is shifted and rotated which changes the cost and information of all child nodes of V_0. Therefore, the rewiring queue $\mathcal{Q}_s$ is emptied after a pose update such that the nodes that are close to the new root node have the highest priority for rewiring. Although this simple update of the root node works in most cases, it can cause trouble if

the distance between the new and the old root node is greater than the rewiring radius r. In this case, the rewiring can fail and some nodes remain child nodes of V_0 even if the edge distance is greater than r. To prevent this, a new node is created at the most recent pose estimate, whenever the distance to the last root node is greater than r. For example, the distance between the new and the old root node can be greater than r when the communication to the localization unit is interrupted.

4.3.5 Path Selection

As stated before, a path that should be traversed by the robot needs to be extracted in every iteration. This path should maximize the utility along the path, hence the trade-off between information and cost.

In [NaderiRajamakiHamalainen15], the RT-RRT* algorithm is used to generate a minimum distance path to a specified goal area. Therefore, all nodes in the goal area are checked for their path length and the node with the shortest path length is chosen as the target node V_t. Then, the target path is extracted by simply adding the parent nodes $p(V_t)$ to the path until the root node is reached, hence $\mathcal{P} = [V_0, \ldots, p(p(V_t)), p(V_t), V_t]$.

In an informative path planning framework, a target area could be specified as the area with the highest entropy in the GMRF belief. Then, an informative path reaching that target area can be extracted from the tree. However, as another potentially longer path can have a higher utility, this path extraction would be suboptimal. Therefore, the maximum utility path is extracted as stated in Eq. (4.7). The utility of each node $U(V_i)$ describes a trade-off between information and cost along the path from the root node V_0 to V_i. Consequently, the maximum utility path can be found by simply extracting the path from the root node to the target node with highest utility $U(V_t)$. This node V_t is kept as the target until the robot is sufficiently close to its position $\boldsymbol{x}(V_t)$. While following the path $\mathcal{P}$ from V_0 to V_t, rewiring is performed which further optimizes the path. Additionally, obstacles might occur which is why the path is extracted after each rewiring iteration. When the target position is reached, the next maximum utility path is extracted as before.

4.3.6 Obstacle Avoidance

If a map of the environment and its free space $\mathcal{X}_{free}$ is known in advance, the presented algorithm generates paths that do not collide with obstacles. This is because, whenever an edge is created, it is checked whether this edge intersects with an obstacle or not. In an exploration scenario, however, obstacles

can suddenly occur which change the free space $\mathcal{X}_{free}$ and thus the paths need to be adapted. Oftentimes, a map of the environment is built while the robot simultaneously localizes itself relative to this map. For that purpose robots are generally equipped with range sensors, that can detect obstacles, such as light detection and ranging (LIDAR) systems or RGB-D cameras which provide both, color and depth images. As especially cameras have a limited field of view, large obstacles are only partially detected which results in many map updates and as a consequence requires fast replanning of the robot's path.

The IG-RT-RRT* algorithm enables fast obstacle avoidance which can be formulated as a rewiring step. Once an obstacle is detected, all edges in a neighborhood of the obstacle are checked for collisions. This is done by iterating over all near nodes of the obstacle where the edge of every node to its parent is checked for intersection with the newly added obstacle. If a collision is detected, the child nodes' utility $U(V_i)$ is set to zero and V_i is pushed to the front of $\mathcal{Q}_r$ meaning that V_i is prioritized for rewiring. Setting the utility to zero is needed in order to connect the child node to any of the neighboring nodes which does not result in a collision, since the new utility will always increase. This replanning step is illustrated in Fig. 4.6, where a robot is equipped with a range sensor that can detect obstacles within a circle segment. In this example, three edges collide with the new obstacle and therefore, the three child nodes (right of the obstacle) are pushed to $\mathcal{Q}_r$. After a rewiring step is performed, the edges of the tree are updated, resulting in collision free paths.

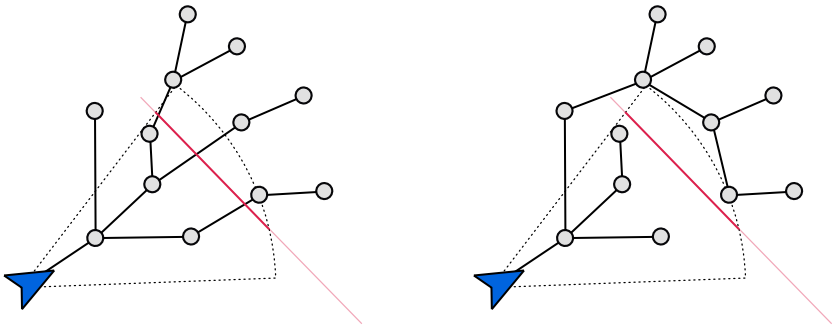


Figure 4.6: Illustration of the obstacle avoidance used in the path planning algorithm. The field of view is given by a circle segment. The obstacle is colored in red.

4.3.7 Implementation Details

The previous sections presented a utility based version of the RT-RRT* algorithm. Before specifying the cost, information and utility formulation, this section reviews some implementation details which are used to reduce the computation time of the algorithm.

The first thing one should notice is that the NearNodes $(\cdot)$ operation in line 8, as well as the distance minimization in line 4, in Algorithm 2 can be computationally demanding. A naive implementation of the NearNodes $(\cdot)$ function could be, for example, to calculate the distance to every node of the tree and extracting the ones which have a distance less than the rewiring radius r. This is clearly unnecessary, as only a few nodes lie within that circle. This problem of finding a set of points that are close to a given point is commonly known as *nearest neighbor search* (NNS) and widely used in the context of machine learning, e.g. for pattern recognition or classification [AbbasifardGhahremaniNaderi14]. A prominent structure for NNS is the KD-Tree which is a space-partitioning data structure that stores the nodes in a binary tree. Once KD-Trees are built, they can be used to efficiently obtain the nearest neighbors of a given node. However, as the number of nodes in the tree $\mathcal{T}$ increases over time, a KD-Tree would have to be dynamically extended which is non-trivial [ProcopiucEtAl03]. Therefore, a simple yet effective storage of nodes is used, similar to the one in [NaderiRajamakiHamalainen15]. The operating space $\mathcal{X}$ is discretized into grid cells which store the nodes. Whenever a NNS is performed, it is checked which grid cells are intersected by the search radius r and the subset of nodes within the intersected cells are reviewed for the NNS. Further, the obstacles are also stored within the grid, such that a collision check is only performed when necessary.

To speed up the collision check between edges of the tree and obstacles, only line segments are used to approximate the obstacle shapes. In contrast to, e.g. an intersection of a line segment and a circle, an intersection of two separate line segments can be identified more quickly. Other commonly used obstacle representations are occupancy grid maps which discretize the space $\mathcal{X}$ into cells where each cell is either occupied or not. Then, a collision check can be done similarly where it is examined whether an edge of the tree intersects with the borders of the occupied grid cell.

4.3.8 Formulation of Cost, Information and Utility

All of the results presented above, can be applied to solve the path planning problem, where any utility function U is to be maximized. In this section, the utility for the IPP is specified as a function of cost and information. Further, it is shown what influence the individual parameters of the utility function have on the tree. For that purpose, the cost, information and utility formulations are covered separately.

Cost Along a Path

In most planning problems, only the cost along a path is considered. This cost function can penalize different quantities such as the energy consumption or control cost. As stated before, the cost along a path is recursively calculated through

$$c\left(V_c\right) = c\left(p\left(V_c\right)\right) + c_{c \to p} \tag{4.9}$$

using the parent cost $c\left(p\left(V_c\right)\right)$ and the cost along an edge $c_{c \to p}$. Therefore, the overall cost along a path is simply the sum of all edge costs. In this work, the edge cost is specified as

$$c_{c \to p} = \omega_{dist} c_{dist} + \omega_{steer} c_{steer} \tag{4.10}$$

where c_{dist} is the distance cost and c_{steer} is the steering cost. Both costs are weighted with the tuning parameters ω_{dist} and ω_{steer}, respectively. The distance cost is given by

$$c_{dist} = \frac{\left\| \boldsymbol{x}\left(V_c\right) - \boldsymbol{x}\left(p\left(V_c\right)\right) \right\|}{d_{nominal}} \tag{4.11}$$

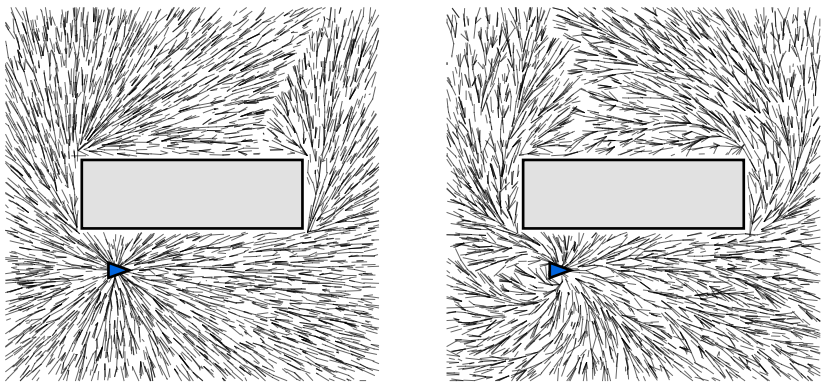
which penalizes the normalized length of an edge. The nominal distance $d_{nominal}$ is the distance between nodes if no rewiring is done, hence the length of the edge created by the steer$(\cdot)$ function. This parameter $d_{nominal}$ could also be included in the weight ω_{dist} but it provides an intuition for the scaling of costs and information if the edge length varies. As the path planning is done in two dimensions, the steering cost is defined as

$$c_{steer} = \frac{1}{c_{dist}} \left(\varphi\left(V_c\right) - \varphi\left(p\left(V_c\right)\right)\right)^2 \tag{4.12}$$

which evaluates the difference of heading angles φ between parent and child node. If $\varphi\left(V_c\right) = \varphi\left(p\left(V_c\right)\right)$ the steering cost is zero since the edge would be on a straight path segment. Additionally, the steering cost is scaled by the reciprocal distance cost. If the distance of the edge is greater than the nominal distance, hence

$c_{dist} > 1$, less control effort is needed to reach the desired heading angle. On the other hand, if two nodes of a path are close together, the same angle difference would require a more aggressive controller. Thus, in some sense, the steering cost is similar to the curvature of a circle segment connecting the two poses.

Figure 4.7 shows the influence of the steering cost. The trees are generated such that the cost defined in Eq. (4.10) is minimized. The left plot depicts the tree if no steering cost is taken into consideration which results in a tree consisting of paths to any point in space with the minimum possible distance given the number of nodes. This, however, has the major drawback that paths are generated which are in the opposite direction of the robot's heading. This is problematic, as many AUVs can not move omnidirectional and thus some paths could not be traversed. The HippoCampus, for example, is only able to move in the direction of its heading. Including the proposed steering cost in Eq. (4.12) solves this problem as points that are behind the robot can only be reached by turning. Although the resulting paths are much smoother as high angle differences between nodes are penalized, kinematic feasibility can not be guaranteed as for many other sampling-based motion planners. However, there are many approaches to generate kinematically feasible trajectories from a finite set of points such as online polynomial trajectory planning [GaoShen16]. Therefore, it is assumed that the generated paths can be followed by an AUV. In Chapter 5, a velocity based motion model in combination with a proportional steering controller is used to successfully follow the planned paths in simulations.



(a) $\omega_{steer} = 0$ (b) $\omega_{steer} = 2$

Figure 4.7: Two trees generated with $n = 10000$ nodes that minimize the path cost. The steering weight is varied while the distance weight is set to $\omega_{dist} = 1$.

Information Along a Path

In the same way as in Eq. (4.9), the information along a path from V_0 to an arbitrary node V_c can be obtained recursively through

$$g\left(V_c\right) = g\left(p\left(V_c\right)\right) + \gamma^{|\mathcal{P}|} g_{c \to p}, \quad 0 < \gamma \leq 1, \tag{4.13}$$

with the modification that the information is discounted using a discount factor γ. If γ is less than one, the information along an edge is worth less with increasing path length $|\mathcal{P}|$ as $\gamma^{|\mathcal{P}|} \to 0$ for $|\mathcal{P}| \to \infty$. Discount factors are commonly used to control the priority of rewards and ensure that the reward along a path is bounded. In this sense, the information can also be seen as a reward that should be maximized along the path. As discussed before, informativeness can be expressed in different metrics such as conditional entropy or mutual information as notion for uncertainty, see Section 4.2. Since the computational cost is an important factor, it has been concluded that the conditional entropy is the choice of metric. Especially in the context of field exploration, uncertainty is not the only source of informativeness, as it could also be of interest to find areas of high concentration in the field. Therefore, the information of a single edge is defined as

$$g_{c \to p} = g_{uncertainty} + \omega_{mean} g_{mean} \tag{4.14}$$

which consists of the uncertainty $g_{uncertainty}$ along an edge as well as the reward g_{mean} of the mean field value which can be scaled using by the weight ω_{mean}. Both pieces of information can be obtained through the GMRF belief which is known to the path planner. The uncertainty reward along an edge is defined as

$$g_{uncertainty} = c_{dist} \frac{\mathcal{H}\left(f_{\boldsymbol{x}(V_c)} | \boldsymbol{f}_{\mathcal{A}}\right) + \mathcal{H}\left(f_{\boldsymbol{x}(p(V_c))} | \boldsymbol{f}_{\mathcal{A}}\right)}{2} \tag{4.15}$$

which represents the mean conditional entropy between random variables $f_{\boldsymbol{x}(V_c)}$ and $f_{\boldsymbol{x}(p(V_c))}$ at the location $\boldsymbol{x}(V_c)$ and the parent location $\boldsymbol{x}(p\left(V_c\right))$, respectively. The entropy is conditioned on the set of taken measurements $\boldsymbol{f}_{\mathcal{A}}$. This mean conditional entropy is scaled by the distance cost c_{dist} as longer edges carry more information about the field. A more intuitive understanding is given by the fact that, with increasing length, more measurements can be taken along an edge and should therefore be additionally rewarded.

When evaluating the uncertainty along a path at planning time t, the conditional entropy of an arbitrary node on the path $\mathcal{H}\left(f_{\boldsymbol{x}(V_i)} | \tilde{\boldsymbol{f}}_{\mathcal{A}}\right)$ depends on all measurements that have been taken up to time t as well as the future measurements which are taken along the path from V_0 to V_i. This set of measurements is given by $\tilde{\boldsymbol{f}}_{\mathcal{A}} = \boldsymbol{f}_{\mathcal{A}}^t \cup \left\{ f_{\boldsymbol{x}(V_j)} | j = 0, 1 \ldots, i-1 \right\}$ which unites all physical measurements $\boldsymbol{f}_{\mathcal{A}}^t$ that have been taken and the simulated measurements $f_{\boldsymbol{x}(V_j)}$ at

the vertex positions $\boldsymbol{x}(V_j)$. Ideally, the entropy of an arbitrary node V_i would be obtained by conditioning the most recent GMRF belief on all future measurements $\left\{ f_{\boldsymbol{x}(V_j)} | j = 0, 1 \ldots, i-1 \right\}$ along the path from V_0 to V_i. However, if the entropy of an edge is calculated using simulated measurements, each node in the tree would maintain its own GMRF belief which has to be conditioned every time the edges of the tree are updated. This is of course unsuitable in terms of computational storage and computation time for an online application, which is why simulated measurements along the path are neglected in the uncertainty formulation. Furthermore, as the selected path is replanned while following it, the path is adapted to the additional measurements such that new information is always taken into account.

The mean information g_{mean} in Eq. (4.14) is used as a reward for field values which can be used to focus the exploration on high concentration areas. The mean information along an edge is defined as

$$g_{mean} = g_{uncertainty} \frac{\mu_{\boldsymbol{x}(V_c)} + \mu_{\boldsymbol{x}(p(V_c))}}{2} \tag{4.16}$$

which is the averaged mean field concentration between node position $\boldsymbol{x}(V_c)$ and parent node position $\boldsymbol{x}(p(V_c))$ multiplied by the uncertainty of the edge. This means that g_{mean} is large if the expected field concentration is high while also being very uncertain about it. If the mean reward is not scaled by the uncertainty, the same reward would be obtained even if many measurements have been taken which could lead to a robot circling a high concentration value which, however, could be only a local maximum. Scaling the mean field concentration by the uncertainty has the effect, that the measurement locations are placed in areas of high concentrations until the uncertainty is sufficiently low which is a balance of exploring the whole operating space and exploiting areas with high field values.

Having defined the information along a path, one could argue to simply maximize the information g to obtain an informative path. Although the reward along a path is bounded due to the discount factor γ, maximizing the information would result in infinitely long paths in the form of cycles. This is because the information g is a monotonically increasing function over the path length and thus the sum of edge rewards is maximized for a closed path and would approach infinity for $\gamma = 1$. Infinitely long paths, however, are unsuitable for the proposed planning algorithm since simulated measurements are neglected. A planned path that crosses the same location multiple times, as in the case of cycles, is not of interest since the information at a specific location decreases drastically after the robot takes a measurement at that location. An additional consequence of cycles is that the PropagateValues$(\cdot)$ function in line 10 of Algorithm 3 leads to an endless loop since the tree does not have any leaf nodes. Thus, the utility is introduced which solves this problem by penalizing infinitely long paths.

Utility Along a Path

In order to generate paths that carry more information with shorter path length, the utility function U is introduced. This function is used to trade-off the information and the cost along a path. Both quantities, c and g are monotonically increasing functions over the path length which is why maximizing g would result in infinitely long paths. Thus, an important requirement for the utility function to be maximized is that it is a decreasing function over the path length $|\mathcal{P}|$. Further, this function should reward the information and penalize the cost. Therefore, the proposed utility function for evaluating the quality of a path from V_0 to an arbitrary node V_c is defined as

$$U(V_c) = \frac{\sum_{V_i \in \mathcal{P}_c} g_{i \to p(i)}}{\left(\sum_{V_i \in \mathcal{P}_c} c_{i \to p(i)}\right)^{\alpha}}, \quad \alpha > 1 \tag{4.17}$$

$$= \frac{g(V_c)}{c(V_c)^{\alpha}} \tag{4.18}$$

which is the ratio of information to cost along the path $\mathcal{P}_c$ connecting the root node V_0 to the node V_c. To ensure a decreasing function, the tuning parameter α is introduced which can be seen as an importance weight of the cost along the path. This tuning parameter needs to be greater than 1 for U to be a valid function used for evaluating paths of the tree $\mathcal{T}$. In order to show, that Eq. (4.18) is indeed a decreasing function over the path length, a simplified scenario is examined.

Suppose, a tree $\mathcal{T}$ is built in a workspace $\mathcal{X}$ where all edges have the same length and only the distance (not the steering) is penalized. Further, the information along all edges is also the same, hence the cost and information between nodes are constant and can be expressed as $g_{i \to p(i)} = G$ and $c_{i \to p(i)} = C$, respectively. This is for example the case when no measurements have been taken, thus the uncertainty in the GMRF belief is the same for every point in space. With that simplification, the utility function can be expressed as a function of the path length $|\mathcal{P}|$, reading

$$U(|\mathcal{P}|) = \frac{\sum_{i=0}^{|\mathcal{P}|} G}{\left(\sum_{j=0}^{|\mathcal{P}|} C\right)^{\alpha}} \tag{4.19}$$

$$= \frac{|\mathcal{P}|\, G}{|\mathcal{P}|^{\alpha}\, C^{\alpha}} = \frac{G}{C^{\alpha}} |\mathcal{P}|^{(1-\alpha)}. \tag{4.20}$$

Taking the derivative with respect to the path length,

$$\frac{\partial U}{\partial |\mathcal{P}|} = (1-\alpha) \frac{G}{C^{\alpha}} |\mathcal{P}|^{-\alpha} < 0 \quad \forall \alpha > 1, \tag{4.21}$$

shows that U is monotonically decreasing for all values of α greater than one.

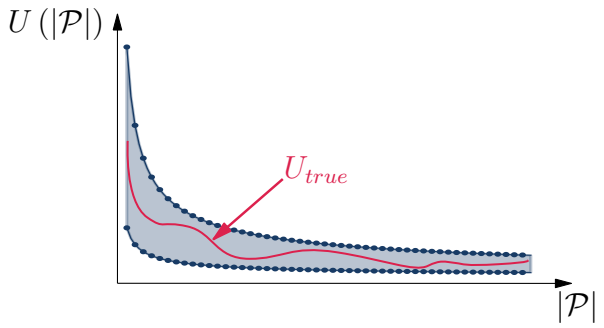


Figure 4.8: Illustration of the bounded utility function. The true utility function lies in a sector which is defined by an upper and a lower bound.

This is of course a simplified scenario since the cost and information between nodes is usually not constant. However, assuming that $G = G_{max}$ is the maximum possible information and $C = C_{min}$ is the minimum cost, Eq. (4.19) provides an upper bound for the true utility function defined in Eq. (4.18). In the same way, a lower bound can be generated which, together with the upper bound, defines a sector that includes the true utility function. Although monotonic decrease can not be guaranteed, it is ensured, that the utility decreases globally with increasing path length, as illustrated in Fig. 4.8. Hereby, the local maxima are important as in some cases, longer paths are favored which makes sense since they are likely to carry more information.

If, as before, only a monotonically increasing cost function is minimized, the resulting path to any point in space would never include a detour as longer paths always increase the costs. When maximizing the information, in contrast, the resulting path would be as long as possible since the information is added up. When using the utility function which includes both, cost and information, longer paths are generated which are ensured to be of finite length. Two exemplary informative trees which are generated using the notion of utility are depicted in Fig. 4.9. The left tree in Fig. 4.9 is generated, rewarding only the conditional entropy which results in many paths that pass the corners since the entropy is large in these areas. Further, it is interesting to see that along the trace where measurements have been taken (colored in dark blue), the paths of $\mathcal{T}$ are separated since crossing that area has a low utility. The right plot in Fig. 4.9 shows a tree that rewards high concentration areas which are colored in yellow. In this case all paths to the upper boundary of the workspace pass through the high concentration area even though this is a detour for some points.

To sum up, this section introduced the notion of utility which is used to trade-off the costs and information along a path. It has been shown that the proposed utility description is a valid function that can be maximized because of its prop-

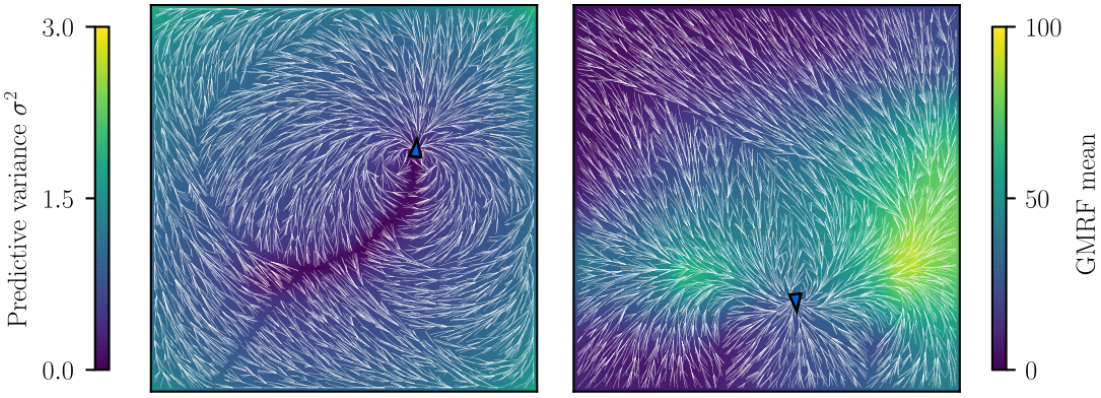


Figure 4.9: Two informative trees maximizing the utility function U, generated with $n = 10000$ nodes. The left plot shows the tree on top of the map of predictive variances of the GMRF belief where dark colors represent areas of low uncertainty. The right plot shows a tree with a mean weight $\omega_{mean} > 0$ on top of the mean GMRF belief with high concentrations depicted in bright colors.

erty to decrease globally with increasing path length. The exponent α can be used to tune the the importance of the cost along the path. For large values of α, the cost is of higher priority which results in trees as depicted in Fig. 4.7. The closer α is to one, the longer are the paths generated by the planning module which can include detours to maximize the information. Sometimes, however, the resulting paths can include cycles if α is too close to one since in this case the information is of high priority. As a consequence, the planned path crosses the same measurement locations multiple times if the uncertainty is high. This is due to the simplification that no measurements are simulated along the path. Therefore, the parameter α needs to be tuned carefully to find the most suitable trade-off between information and cost along a path.

Chapter 5

Simulations of Environmental Field Exploration

In this chapter the performance of the proposed Information Gathering RT-RRT* (IG-RT-RRT*) algorithm is analyzed in simulations of environmental field explorations using an autonomous robot. First, the path planning module is analyzed regarding its sensitivity to changes in the most influential tuning parameters. Then, a case study is conducted which examines the two scenarios of source localization and exploration in cluttered environments. These simulations are carried out in a 2D simulation environment using a velocity based motion model. In order to show the functionality for an underwater exploration scenario, the presented algorithms are further analyzed in a more sophisticated physics based 3D simulation environment which also takes into account effects such as buoyancy.

5.1 2D Simulation Environment

In this section, a C++ simulation environment is used for a comprehensive analysis of the exploration performance. In Chapter 3, the RMSE was used to measure the performance of different belief models. It has been shown, that GMRFs are highly suitable for representing environmental fields which is why all of the following results are based on a GMRF belief. In order to measure the performance of the path planning module, a different metric is used. In the case of an exploration it is of interest how the uncertainty in the field belief is reduced over time. Therefore, the sum of predictive variances of the conditioned GMRF belief is used as performance metric, reading

$$V = \sum_{i=1}^{n} \sigma^2_{f_i|\mathcal{A}}, \tag{5.1}$$

where n denotes the number of GMRF variables. Equation 5.1 provides a scaled version of the fields entropy and is thus suitable for evaluating the exploration performance. Furthermore, the control cost is an important factor since the power supply of AUVs is usually limited. To define the control cost, a mathematical model of the robots dynamics has to be defined first. In this work, a velocity based motion model is used which only takes into account the kinematics of a robot. Hereby, the robots state is given by a 2D pose consisting of the robots position in an inertial frame as well as its orientation φ. The discrete dynamics model is given by

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \varphi_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \varphi_t \end{bmatrix} + \begin{bmatrix} -\frac{v}{u_t}\sin(\varphi_t) + \frac{v}{u_t}\sin(\varphi_t + u_t\Delta t) \\ \frac{v}{u_t}\cos(\varphi_t) - \frac{v}{u_t}\cos(\varphi_t + u_t\Delta t) \\ u_t\Delta t \end{bmatrix} \tag{5.2}$$

which describes how the state of the system evolves over a time Δt. It is assumed that the robot moves with constant velocity v while the angular velocity is given by the control input $u_t = \omega$. Note, that this model assumes that the robot moves on a circle of radius $r = |v/\omega|$ if both velocities are constant. For a detailed derivation of the mathematical model, the reader may refer to [ThrunBurgardFox05]. The control input u_t is used for tracking of the path that is provided by the planning module. Given a sequence of nodes, that should be traversed by the robot, u_t is used to steer to the closest node which can be achieved by application of the proportional controller

$$u_t = -K_p e = -K_p \left(\tan^{-1}\left(\frac{y_{goal} - y_t}{x_{goal} - x_t} \right) - \varphi_t \right) \tag{5.3}$$

which drives the angle difference e to zero, as illustrated in Fig. 5.1. The commanded angular velocity can be tuned with the proportional gain K_p.

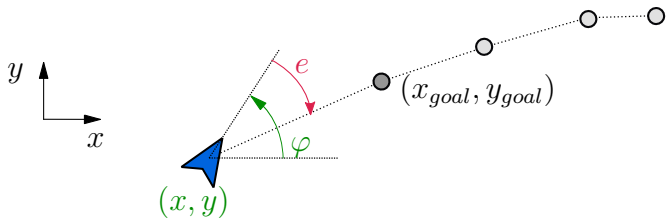


Figure 5.1: Illustration of the steering controller which drives the angle difference e to zero. The planned path is depicted as a sequence of nodes.

Since the velocity v is assumed to be constant, the only control cost that can be reduced is the steering given by u_t which is why the second metric used for

Table 5.1: The default set of parameters used in the path planning module.

Parameter	Symbol	Value
Field dimensions	$\ell_x \times \ell_y$	10×10 m
Velocity	v	$0.5\frac{\text{m}}{\text{s}}$
Nominal edge length	$d_{nominal}$	0.3m
Rewire radius	r	0.4m
Distance weight	ω_{dist}	1
Steering weight	ω_{steer}	2
Mean weight	ω_{mean}	0
Discount factor	γ	0.99
Cost exponent	α	1.4
Number of nodes	n_{nodes}	6000

evaluating the exploration performance is defined as

$$\mathcal{U} = \frac{1}{N} \sum_{t=0}^{N} u_t^2. \tag{5.4}$$

This provides a measure for the mean steering cost over the entire exploration that is done in N time steps.

In order to prove the real time capabilities of the proposed Information Gathering RT-RRT* algorithm, the rewiring time is limited to 100 ms meaning that the path is updated at a frequency of 10 Hz. Further, the root of the tree as well as the GMRF belief are also updated every 100 ms. The default parameters used in the subsequent sections are summarized in Tab. 5.1. The hyper-parameters of the GMRF belief are the same as presented in Chapter 3.

Since points are randomly sampled in the tree's expansion, the planning behavior is not deterministic. Therefore, each simulation is repeated 30 times and the median and the interquartile range (IQR) are used to evaluate the performance of all simulation series. The following simulations in the C++ environment are carried out on a computer system with a Dualcore-CPU 'i5-7200U' with 2.5 GHz and 8GB RAM.

5.1.1 Parameter Variations

This section provides an insight into two main tuning parameters of the Information Gathering RT-RRT* algorithm. The performance of the overall exploration is analyzed with respect to changes in the tuning parameters. The metrics used for evaluating the performance are the predictive variance sum V in Eq. (5.1),

the mean control cost $\mathcal{U}$ in Eq. (5.4) and the time needed to rewire the entire tree.

Number of Tree Nodes

As stated before, the RRT* algorithm is said to be asymptotically optimal with respect to a cost or utility function as $n_{nodes} \to \infty$. It therefore stands to reason that more nodes will generally improve the exploration performance. However, as the number of nodes increases, less parts of the tree can be rewired in limited time and thus suboptimal paths can be generated. Consequently, the trade-off between the possibility of better paths and the quality of the resulting paths is analyzed based on the number of nodes in the planning tree.

Figure 5.2 shows the mean GMRF belief as well as the traversed paths for different numbers of nodes in the tree. A general observation is that the field is represented well in all three cases even though not the whole workspace has been covered which shows that GMRFs are advantageous for making predictions at unobserved locations. Further, it is clearly noticeable that the borders of the field are preferred in the beginning of the exploration which is due to the fact that the entropy is large in these areas. In all three explorations, the traversed "global" path is very similar since the robot first explores the boundaries of the space and then plans its path within the area that is enclosed by the outer path. The main difference can be seen in the smoothness of the paths. As more nodes are added to the tree, more and especially smoother paths are available since a larger area of the workspace is covered by nodes. In the case of $n_{nodes} = 2000$, it can be seen that the path is much more curving and that the robot is not able to move into the corners which is due to the lack of nodes at the boundaries of the space. With an increasing number of nodes, the robot tends to move on straighter lines. Therefore, it can be concluded that the generated paths are mostly straight lines for $n_{nodes} \to \infty$ since this minimizes the steering cost.

The median and the IQR of the predictive variance sum V of the GMRF belief is plotted over the exploration time in Fig. 5.3. The performance of the Information Gathering RT-RRT* algorithm is benchmarked against two baseline policies which are given by a random walk and a myopic planning strategy. A robot that follows a random walk policy moves with constant velocity v and samples a random steering command u_t from a zero mean normal distribution. The myopic planner generates 50 different paths with a steering command in a range $u_t \in [u_{min}, u_{max}]$ and executes the path which maximizes the conditional entropy. The myopic planning strategy is deterministic since the sequence of control commands does not change in the simulation series. Figure 5.3 shows that the Information Gathering RT-RRT* algorithm is superior to the baseline policies since the predictive variance sum decays faster and has a notable lower

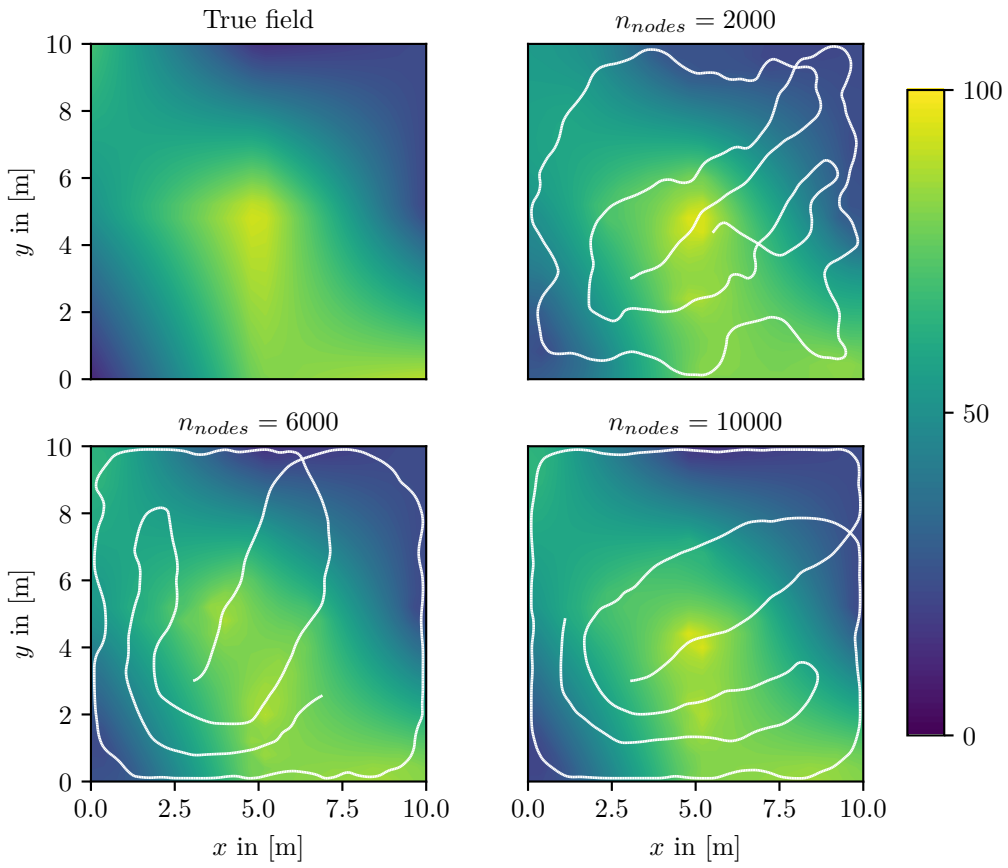


Figure 5.2: True field and mean GMRF beliefs after an exploration time of 100 seconds for different numbers of tree nodes. The robot's path is depicted in white. The robot's initial position is given by $x = 3$m and $y = 3$m.

uncertainty in terms of V after 250 seconds. Furthermore, it is interesting to see that the predictive variance sum decreases with additional nodes in the tree except for the tree that is generated using $n_{nodes} = 20000$. This is due to the fact that only parts of the tree can be rewired in 100 ms which results in suboptimal paths.

To further analyze this effect, the time needed for a rewiring of the entire tree is shown in Fig. 5.4(b). It can be seen that the tree can not be entirely rewired within 100 ms for $n_{nodes} \geq 10000$. Moreover, the increase in the rewiring time is nonlinear which is because more nodes have to be processed in a single rewiring step as more nodes lie within the rewiring radius r. It is therefore to be expected that the rewiring time of the entire tree has a quadratic complexity since both, the number of nodes to be processed and the number of nodes checked for rewiring, increase linearly. An additional effect of a not entirely rewired tree is that the

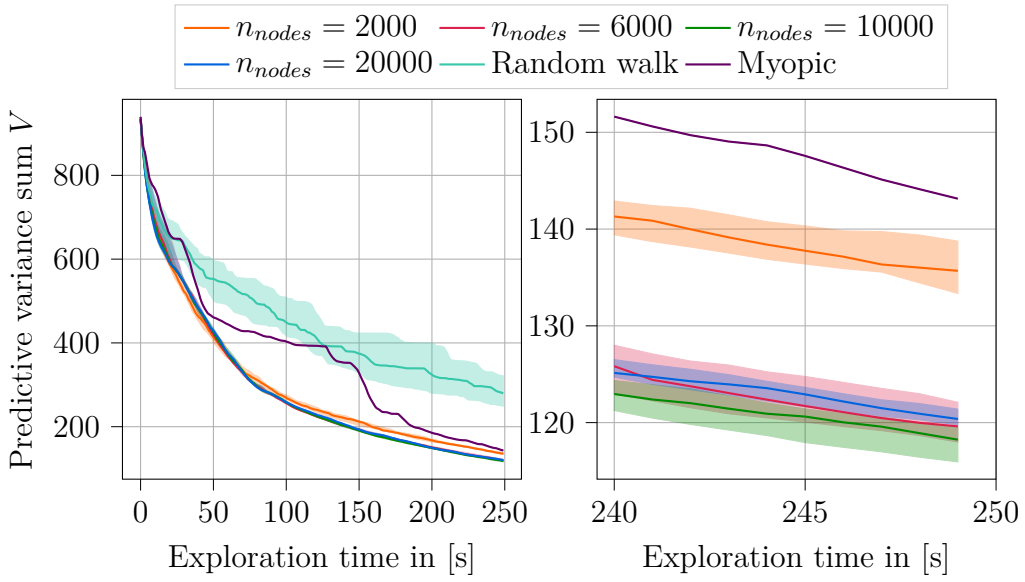


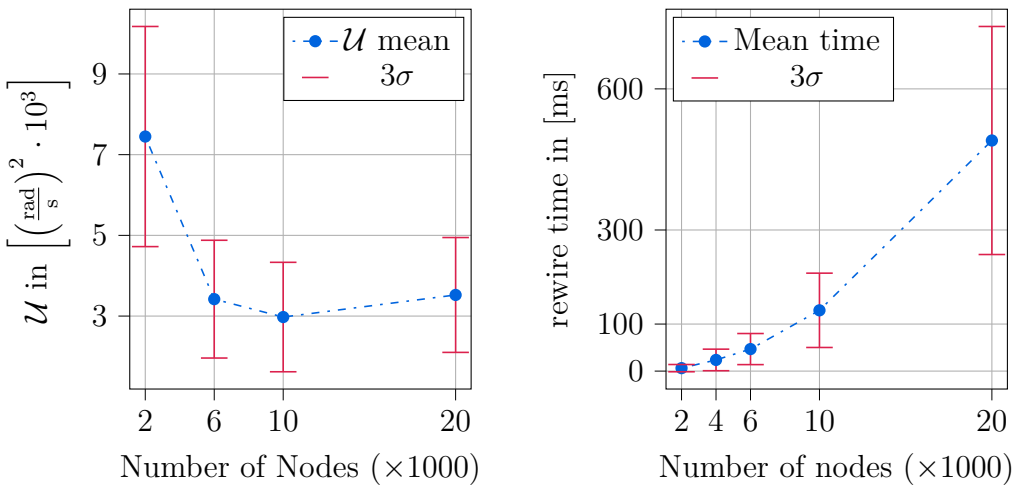
Figure 5.3: Median and IQR of the predictive variance sum V for different numbers of tree nodes compared to the baseline policies.

mean control cost of the exploration has a minimum between $n_{nodes} = 6000$ and $n_{nodes} = 20000$, as it can be seen in Fig. 5.4(a). This is because the generated paths are not sufficiently optimized since the rewiring time is not enough to rewire all edges of the tree. This can be counteracted by increasing the rewiring time which, however, results in a lower update rate of the path and thus less adaptability to external influences such as deviations from the planned path or suddenly occurring obstacles.

The conclusion that can be drawn from this series of simulations is that an increase in the number of tree nodes does not necessarily improve the performance of the exploration. The best results have been obtained in the case $n_{nodes} = 10000$ although the rewiring time of 100 ms is not sufficient to rewire the entire tree. Since the results for $n_{nodes} = 6000$ are already close to the best results and the entire tree can be rewired in the limited time, this configuration is chosen for the subsequent simulations.

The Discount Factor

The discount factor γ is used to weigh the importance of immediate rewards in the path planning algorithm. In the information formulation in Eq. (4.14), the discount factor is raised to the power of the path length such that the information along an edge is more valuable if the edge is closer to the current position of the robot. If γ is small, future rewards are not particularly considered and shorter



(a) Control cost of the exploration

(b) Rewiring time of the entire tree

Figure 5.4: Comparison of the control cost and the rewiring time for an increasing number of nodes.

paths are preferred. On the other hand, if γ is close to one, immediate rewards and future rewards are valued similar which can result in an exploration behavior where the robot is moving greedily to areas of high uncertainty. The influence of variations of the discount factor γ is examined in this simulation series.

The map of the GMRF predictive variance σ^2 as well as the robot's traversed path are shown in Fig. 5.5 for different discount factors. For a parameter $\gamma = 0.6$ it can be seen that the robot can get stuck in a local minimum where the robot is only exploring a part of the space. In order to leave an area that is enclosed by the robot's own path, hence previously visited locations, the robot needs to cross an area which has a low uncertainty and thus a low reward. Since crossing a low uncertainty area has a low utility, the resulting paths avoid intersections with the previously traversed path even if more rewards are expected behind. Moreover, the selected paths are notably shorter for lower discount factors which is due to the fact that the utility of longer paths is lower since future information is less considered. As a consequence of shorter paths, the robot changes its orientation more frequently as the next path is often chosen such that the robot moves in another direction where the entropy is higher. For a discount factor of $\gamma = 0.99$, the generated paths are longer and also cross preciously visited areas if sufficient reward is expected in the future, as it can be seen in the bottom left of Fig. 5.5. For a discount factor $\gamma = 1$, all rewards are equally valued. This has the effect that long paths are generated which more often cross already visited areas since a single uninformative edge is compensated by many future rewards.

Figure 5.6 shows the predictive variance sum for the different discount config-

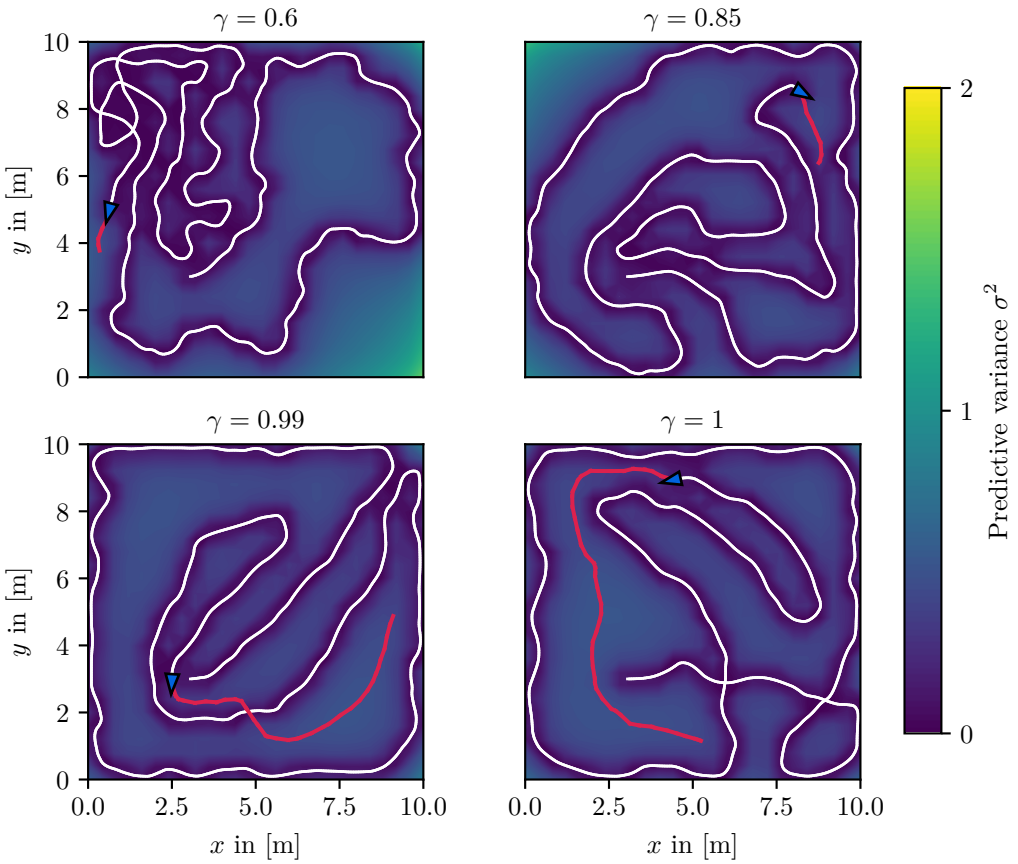


Figure 5.5: Predictive variance maps of GMRF beliefs after an exploration time of 150 seconds for different discount factors γ. The robot's path is depicted in white and the most recent planned path is colored in red.

urations. It can be observed that the predictive variance sum decreases with increasing discount factor which is, however, saturated as a discount factor of $\gamma = 1$ does not improve the exploration performance. This is likely for the reason that local areas are not sufficiently explored before visiting another region in the operating space. When changing the focus of the exploration to another area, previously visited locations are revisited which is a partial waste of information. Consequently, a discount factor of $\gamma = 0.99$ is preferable for the exploration since it achieves the highest uncertainty decrease.

5.1.2 Case Study

In the previous simulations it has been shown that the proposed Information Gathering RT-RRT* algorithm works well when the goal of an exploration is to

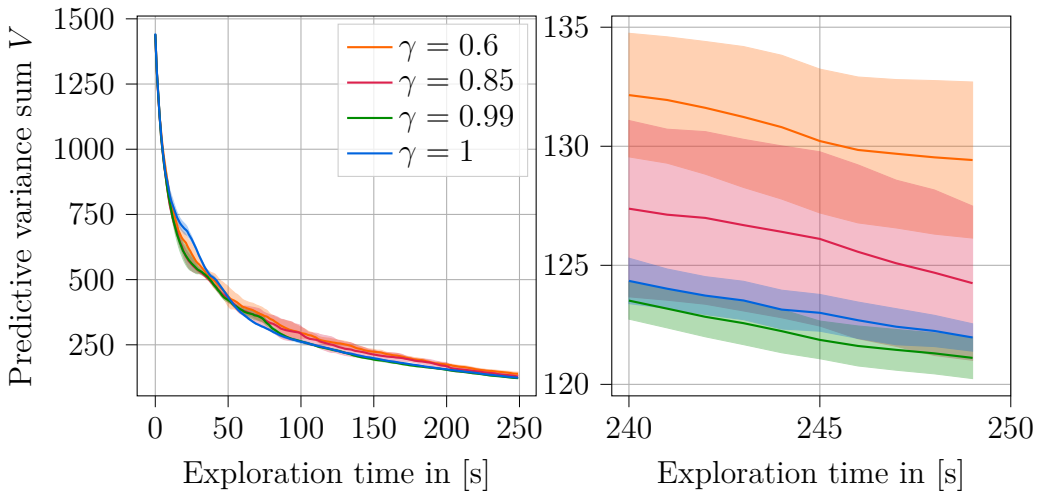


Figure 5.6: Median and IQR of V over the exploration time for varied discount factors γ.

reduce the uncertainty in the field belief. As stated in the problem statement, the path planning module should be flexible in the sense that it is further able to generate paths for a user defined mission goal. Therefore, two challenging special cases where the planning module performs particularly well are covered in this section.

Case I: Source Localization

Consider the scenario of a pipeline leakage below sea level where a hazardous pollutant flows into the ocean. Due to the ocean current, the pollutant is distributed in a large area with multiple local maxima where the pollutant accumulates due to special current conditions. In order to find the leakage as fast as possible and prevent dramatic consequences for the environment, an AUV is deployed which makes use of the field exploration algorithm presented in this thesis.

In order to localize the source of the pollutant, a map of the pollutant distribution is needed which is very accurate in high concentration areas. Therefore, the goal is to reduce the uncertainty in these areas rather than reducing the uncertainty in the whole field belief. This can be achieved by using the tuning parameter ω_{mean}, introduced in Eq. (4.14), which rewards the mean field concentration along an edge of the planning tree. The mean reward along an edge is multiplied by the uncertainty of an edge because a high field concentration should not be additionally rewarded if the uncertainty is already sufficiently low. To analyze the performance of the exploration algorithm for the special case of a source localization, a field is generated which consists of multiple peak concentrations which

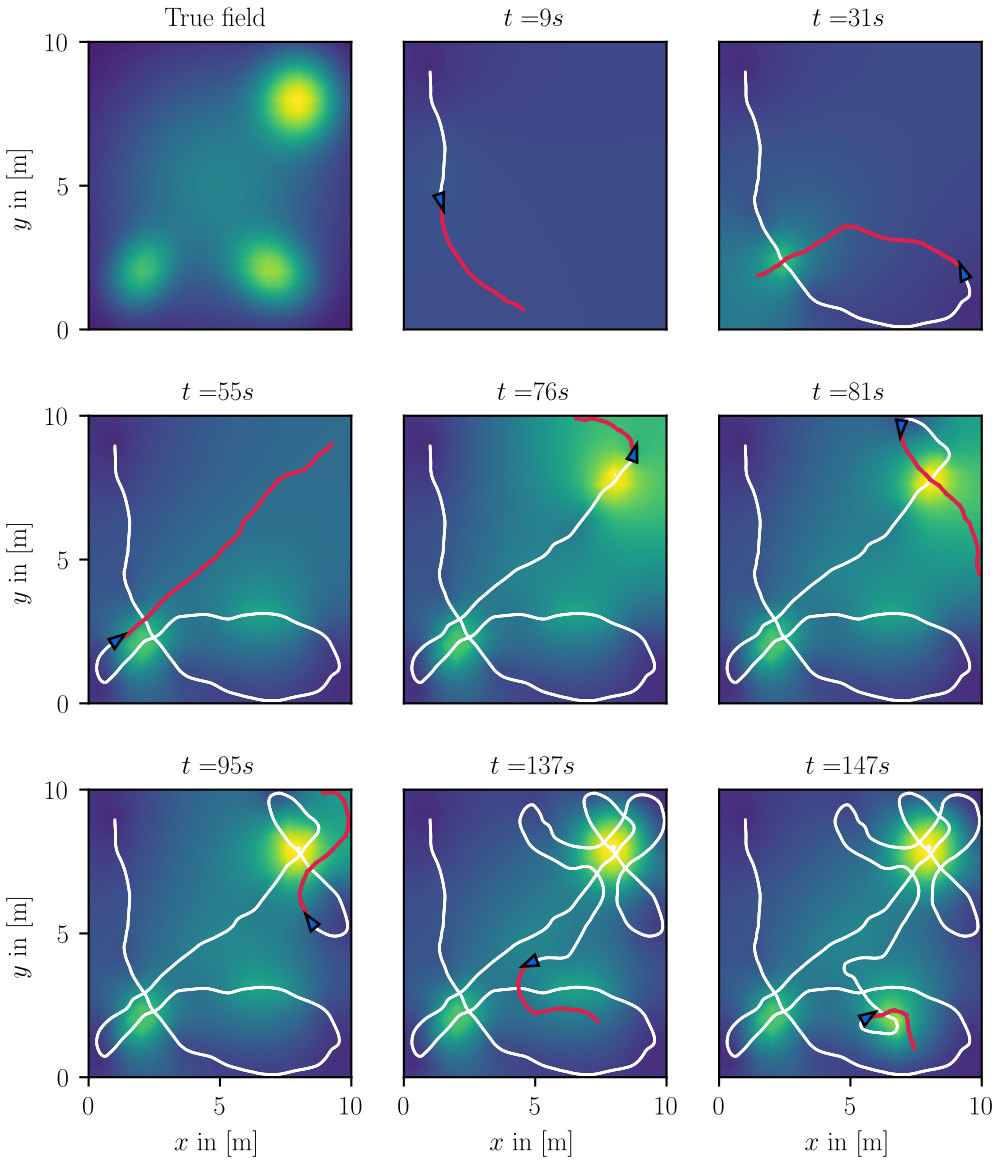


Figure 5.7: An exploration scenario where the robot is supposed to explore areas with a high field concentration. The robot's traversed path (colored in white) is plotted on top of the mean GMRF belief. The robot's planned path is depicted in red.

represent local maxima of the field. In Fig. 5.7, for example, a field consisting of four local maxima which are modeled by Gaussian distributions is to be explored. This field models the aforementioned scenario of a single source and multiple local maxima which occur due to current conditions. The source of the field concentration is located in the upper right corner at the location $(x, y) = (8, 8)$m and has the highest field value. The mean GMRF belief as well as the planned paths

are shown at different exploration times t. It can be clearly seen that the focus of the exploration is on the peak distributions where most of the time is spent to explore the highest field concentration and thus the source of the pollutant. After finding the first local maximum between $t = 9$s and $t = 31$s, the robot wants to revisit that area as the highest concentration of the mean GMRF belief lies in that area. After taking additional measurements to reduce the uncertainty, the exploration is guided towards the source at $t = 55$s since the product of the expected concentration and uncertainty is high in that area. Having located the source of the pollutant at time $t = 76$s the robot spends approximately 50 seconds to explore the surrounding of the source, before exploring other regions of the operating space.

To evaluate the performance of the source localization scenario, a modified metric of the RMSE is used. Since the accuracy of high concentration areas is of greater importance than the error in the entire field, the weighted RMSE (wRMSE) is examined. The wRMSE is computed by weighting the squared errors $(\mu(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2$ between the mean GMRF belief and the true field, reading

$$\text{wRMSE} = \sqrt{\frac{\sum_{i=1}^{n} w_i \left(\mu(\boldsymbol{x}_i) - f(\boldsymbol{x}_i)\right)^2}{n}}, \quad w_i = \frac{f(\boldsymbol{x}_i) - f_{min}}{f_{max} - f_{min}}. \tag{5.5}$$

The weights w_i are normalized to the difference between the maximum field value f_{max} and the minimum field value f_{min} which results in higher weights for high field concentrations $f(\boldsymbol{x}_i)$. Figure 5.8 shows the median and the IQR of the wRMSE for an exploration of fields which consist of four maxima that are randomly placed in the operating space. The proposed exploration algorithm is compared to itself for a mean weight $\omega_{mean} = 0$ and $\omega_{mean} = 10$. In the case of $\omega_{mean} = 0$, paths are generated to achieve an uncertainty reduction in the entire field. As it can be seen in Fig. 5.8, the accuracy of the field belief increases in terms of the wRMSE if the mean GMRF belief is considered in the planning step by using $\omega_{mean} > 0$. Besides from a higher accuracy after an exploration time of 250 seconds, the wRMSE also drops notably faster which is likely because the four local maxima of the field are identified faster in the case of $\omega_{mean} = 10$ compared to $\omega_{mean} = 0$. Therefore, the results show that the proposed planning algorithm is well applicable to a source localization since the resulting map of the environmental field provides a high accuracy in the areas of interest.

Case II: Exploration in Cluttered Environments

In this special case, it is shown that the proposed exploration algorithm can also be used for the exploration of confined environments with many obstacles as in the case of an underwater cave. In [MalliosEtAl14], for example, an AUV is deployed to explore and map a cave complex using simultaneous localization and

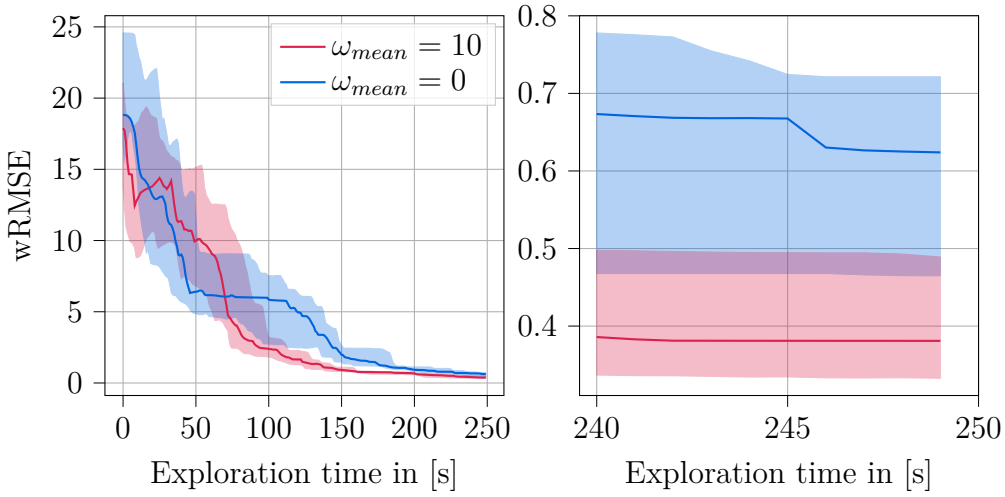


Figure 5.8: Median and IQR of the weighted RMSE over the exploration time for for different tuning parameters ω_{mean}.

mapping (SLAM). It is stated that guidance by a human diver is needed due to the caves' spatial complexity. Therefore, the exploration effort could be greatly reduced by using an autonomous robot that is equipped with an obstacle aware path planning module since external guidance is not required.

In the following simulation, the exploration of a spatially constrained environment is examined where the environment is initially unknown. The robot is equipped with a range sensor that can detect obstacles within a limited field of view. Whenever a new obstacle is detected, the informative path needs to be replanned in order to avoid collisions. It is assumed that the field distribution is independent of the obstacles meaning that the true field can be represented by a smooth function. This function is continuous even in the presence of wall obstacles. This assumption is valid, for example, if the obstacles are given by fishing nets and the field that is to be mapped is a temperature distribution. Figure 5.9 shows the predictive variance of the GMRF belief as well as the traversed path for an exemplary exploration of an environmental field which includes many wall obstacles resulting in a maze like environment. The first observation that can be made is that a global coverage of the work space is achieved since the robot does not get stuck in confined parts of the environment. This is due to the long paths that are generated by the planning module. Longer planning horizons generally allow to early guide the exploration to areas of high uncertainty and thus, achieve global coverage of the field. Secondly, the traversed path is collision free which shows that the adaptive planner is able to quickly react to external influences in form of suddenly occurring obstacles. As a consequence of the global coverage, the robot is further able to also detect all obstacles in the environment. However, there are also some situations in the exploration which could be improved: After

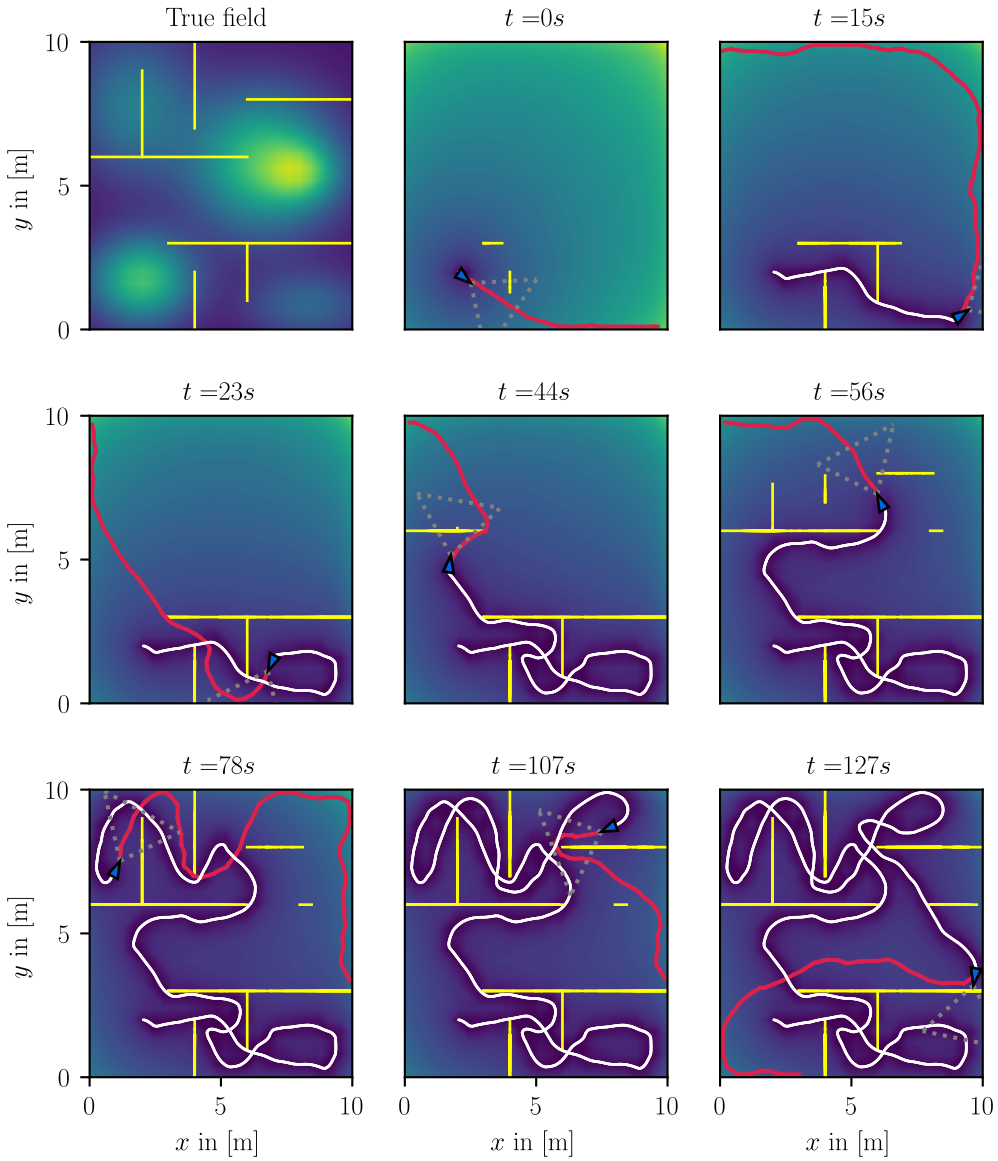


Figure 5.9: An exploration scenario where the robot explores an initially unknown environment with many obstacles (colored in yellow). The first plot shows the true field and the unknown locations of obstacles. The remaining plots show the predictive variance of the GMRF belief. The robot is equipped with a range sensor that can detect obstacles within a limited field of view (dashed lines).

approximately 100 seconds of exploration time, the robot is located in the upper right corner of the field and has to replan its path because of a new obstacle. Since the discount factor is chosen as $\gamma = 0.99$, immediate rewards only play a minor role which is why the robot leaves the confined area in the upper right

corner of the field without sufficiently exploring it. This can be seen in the plots at time $t = 107\,\mathrm{s}$ and $t = 127\,\mathrm{s}$. Therefore, the robot has to revisit that area at some point in time when the uncertainty in the entire field belief is sufficiently low. Reducing the discount factor would result in an exploration behavior where the robot first collects measurements in the confined areas before guiding the exploration to other parts of the environment. However, this could also lead to the robot being stuck in a local minimum, as it has been shown in the analysis of the discount factor in Section 5.1.1. An illustration of the change of the planning tree $\mathcal{T}$ is provided in Fig. 5.10, where an obstacle is added at time $t = 0\mathrm{s}$ and the tree is rewired for 100 ms. All paths in the planning tree are collision free after the rewiring process such that a new goal path to the target location can be selected. The paths that are not intersected by the obstacle remain the same.

To further analyze the performance of the IG-RT-RRT* planning algorithm in cluttered environments, another simulation series is performed where the obstacles are known in advance. In this way, it is easier to compare the performance of the exploration algorithm to the baseline policies which are given by a random walk policy and a myopic planner. Figure 5.11 shows the median and the IQR of the predictive variance sum V after 30 simulations. It is clearly notable that the IG-RT-RRT* algorithm achieves a considerably higher coverage of the field than the baseline policies since there is a large gap between the different mean predictive variance sums after an exploration time of 250 seconds. This is because

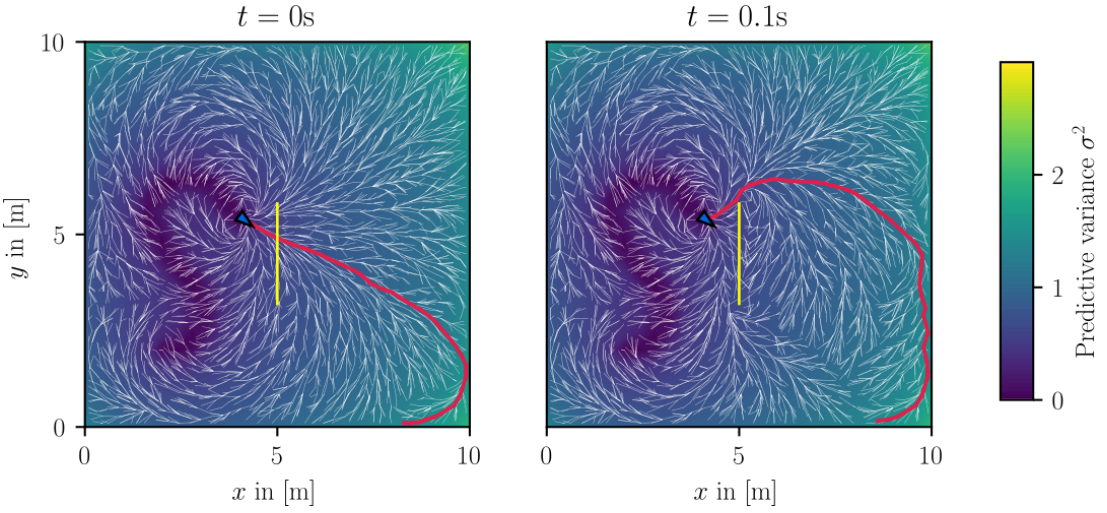


Figure 5.10: Illustration of the change in the planning tree $\mathcal{T}$ when an obstacle (colored in yellow) is detected. The entire tree consisting of 6000 nodes can be rewired within 100 ms. The utility maximizing path is colored in red.

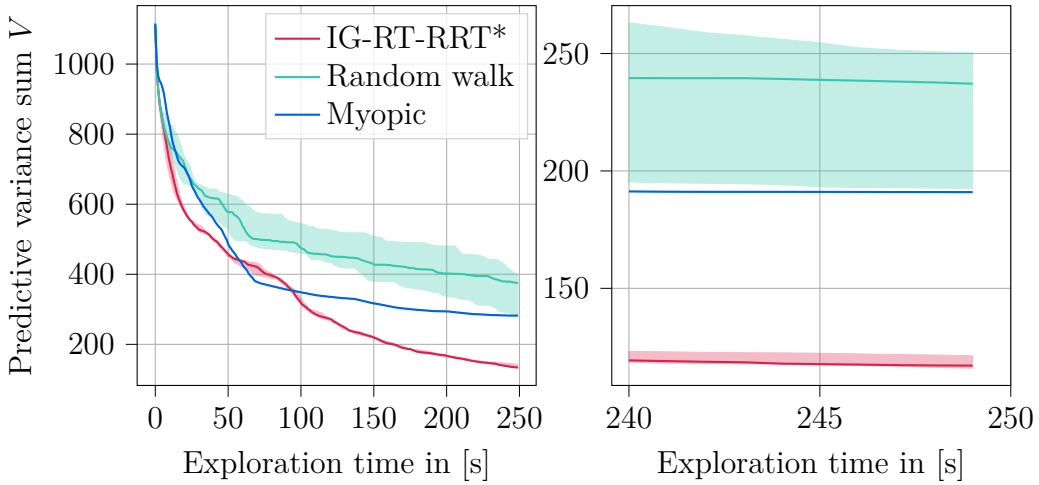


Figure 5.11: Median and IQR of the predictive variance sum over the exploration time for different planning strategies in a cluttered environment.

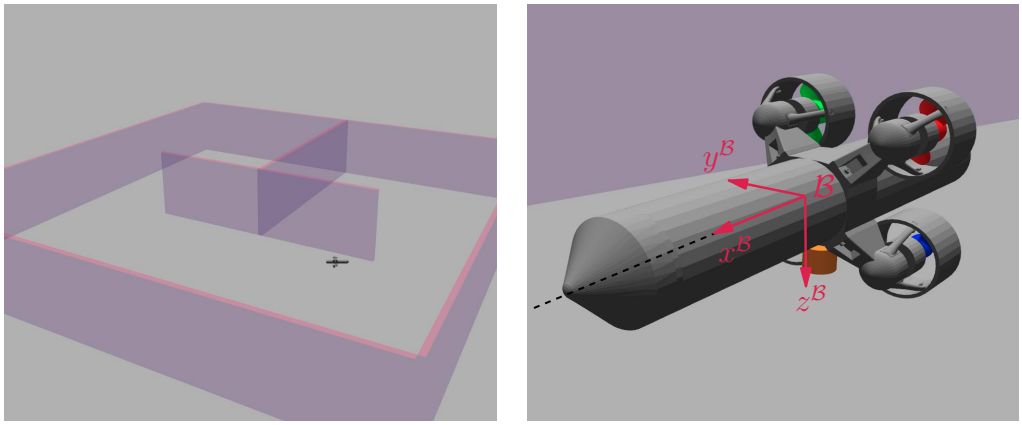
the random walk is not able to explore the entire space and the myopic planner gets stuck in a local minimum.

5.2 Gazebo Simulation Environment

It has been shown in the previous simulations that the proposed algorithm is well applicable to environmental field explorations including the special cases of source localization and exploration in cluttered environments. The underlying model of the robot, however, was a velocity based motion model which only takes into account the kinematics of a robot. This section covers the simulation of a field exploration in a 3D physics based simulation environment called Gazebo. Gazebo is a well-designed simulator which finds vast application in the simulation of robotic systems [KoenigHoward04]. Besides from the robot's kinematics, Gazebo also allows to simulate the dynamics of the robot as well as external forces such as buoyancy forces which have been neglected in the 2D simulation. The goal of this section is to show that the exploration algorithm can also be applied to robotic systems that are more complex than a velocity based motion model.

In [Hastedt19], a model of the HippoCampus μAUV has been integrated in the Gazebo simulation environment. This sophisticated dynamics model also takes into account external forces such as the hydrodynamic forces of the water and models the four actuators of the HippoCampus. Furthermore, an attitude control unit is provided which enables tracking of desired attitudes $\boldsymbol{\Theta}_{des} = [\phi, \theta, \psi]^T$ of the robot. This attitude describes the desired orientation of the robot in terms of the roll angle ϕ around the body axis $x^{\mathcal{B}}$, the pitch angle θ around $y^{\mathcal{B}}$, and

the yaw angle ψ around $z^{\mathcal{B}}$. The local body-fixed coordinate system $\mathcal{B}$ of the HippoCampus is illustrated in Fig. 5.12(b). The control inputs which achieve tracking of desired attitudes are the forces and moments at the four thruster motors that are distributed around the HippoCampus. For a more detailed insight to the control law of the HippoCampus, the reader is referred to [DueckerEtAl20]. Using the control law, the desired pitch angle θ, for example, can be used to control the z position (depth) of the robot. On the other hand, a desired yaw angle ψ can be imposed which enables tracking of a 2D path since changing the yaw orientation at constant roll and pitch angle results in the HippoCampus steering in the $x^{\mathcal{B}}$-$y^{\mathcal{B}}$ plane. Therefore, the path tracking in the Gazebo simulation



(a) Gazebo world. (b) HippoCampus model.

Figure 5.12: The Gazebo simulation environment which is used for validating the exploration algorithm. The Gazebo world can be seen in the left picture and the HippoCampus model in the right picture, respectively.

is done in the following way. As before, the closest node in the path $\mathcal{P}$ is extracted and set as goal position with coordinates (x_G, y_G). Then, as shown in Fig. 5.1, the relative angle between the current yaw orientation of the HippoCampus and the line connecting the origin of $\mathcal{B}$ to the goal point is set as desired yaw angle. This is expressed as

$$\psi = \tan^{-1}\left(\frac{y_G - y_H}{x_G - x_H}\right) \tag{5.6}$$

where (x_H, y_H) denotes the position of the HippoCampus in an inertial frame. This desired yaw angle is passed to the attitude control unit which generates the actuator commands of the four thruster motors. In contrast to the robot in the previous 2D simulation environment, the HippoCampus does not move at constant velocity but at constant thrust. This means that the HippoCampus moves faster when it has to change its orientation since the steering commands generate additional thrust. Moreover, a depth control unit is used which achieves that the

robot operates at a desired water depth. This is needed since the gravitational force acting on the robot does not balance the buoyancy force which would result in the HippoCampus sinking to the ground.

In the following, a simulation of an environmental field exploration in a confined environment is analyzed. In order to show, that the obstacle avoidance also works in a more realistic simulation, a T-shaped obstacle is placed in the environment, as it can be seen in Fig. 5.12(a). The goal of the exploration is to reduce the uncertainty in the GMRF belief such that better conclusions about the true field can be drawn. Figure 5.13 shows the traversed path of the HippoCampus on top of the predictive variance of the GMRF belief. It can be seen that the path has the same characteristics as before, where the boundaries of the space are favored in the beginning. More precisely, the exploration is initially guided to the upper right corner of the workspace since the conditional entropy has its maximum there. This is because the upper right corner is furthest away from the initial position of the HippoCampus where measurements have been taken. Further, it can be observed that the robot is able to avoid collisions with the wall obstacles which once more shows that the proposed algorithm can be applied to exploration in cluttered environments. The exploration of this field was repeated 15 times in simulations, with no collisions occurring. The median and IQR of the predictive variance sum are depicted in Fig. 5.14. This predictive variance sum cannot be directly compared with the results from the 2D simulation since the

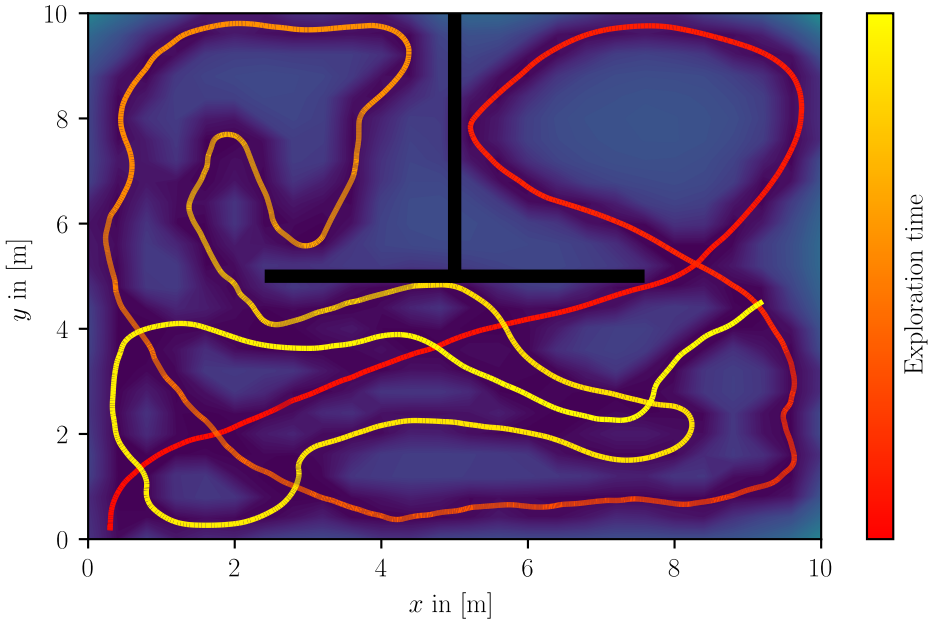


Figure 5.13: The path of the HippoCampus μAUV in an exploration of a confined environment where obstacles are colored in black.

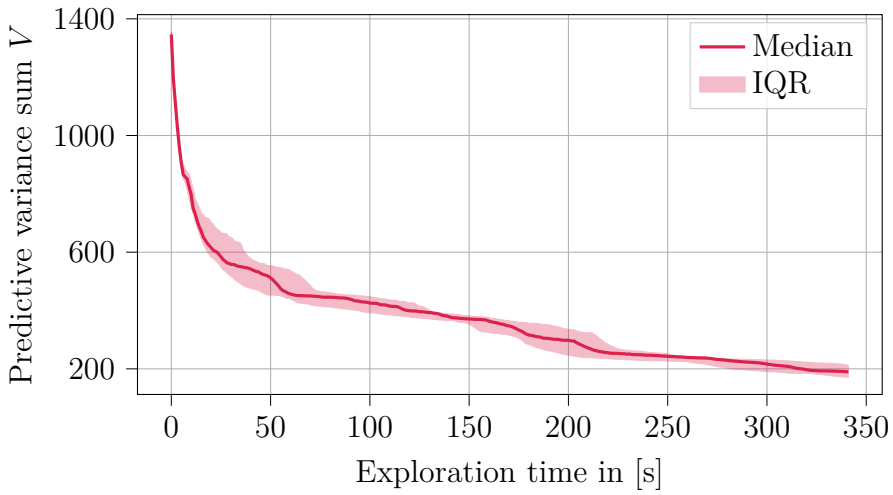


Figure 5.14: Median and IQR of the predictive variance sum after 15 simulations in the Gazebo environment.

velocity of the HippoCampus is not constant. However, it can be seen that V is also monotonically decreasing which speaks for a global coverage of the work space.

In [Horst21], it has been shown that the results from a simulation of the HippoCampus in the Gazebo environment exhibit similar behavior as in a real world experiment. Therefore, from this last simulation series of the proposed exploration algorithm in the Gazebo environment, it appears that the Information Gathering RT-RRT* algorithm can be promisingly applied to complex robotic systems such as the HippoCampus μAUV. The only requirement for an application of the exploration algorithm is a control unit which can track a goal point reliably. However, it should be mentioned that all simulations of the path planner have been done using the ground truth state of the robot without any uncertainty in the robot's pose. Nonetheless, it can be expected that the proposed algorithm would also work in the presence of pose uncertainty because of the advantage of fast replanning.

Chapter 6

Summary and Outlook

6.1 Summary

This thesis presents an exploration algorithm which can be used to explore environmental fields with an autonomous robot. The work of this thesis is two-folded and can be divided into two main modules. For the readers convenience, the requirements of these two modules which have been introduced in the problem statement are listed again subsequently.

Module I: Field Belief The first part of this thesis covers the development of a field belief algorithm which allows to model two-dimensional environmental fields. Besides from a good representation of the true field, additional requirements of the field belief have to be met:

- Limited computational complexity and storage of the algorithm in order to be executed on computationally constrained platforms
- Sequential update of the field belief which allows for online mapping of environmental fields
- Uncertainty measure in the field belief which enables a robust state estimation

Module II: Informative Path Planner The second module of the overall exploration algorithm is the informative path planning module which generates paths of potential measurement locations that contribute to a mission specific goal. This path planner is subject to the following requirements:

- Anytime motion planner which quickly identifies a path whose quality improves with increasing computation time

- It should offer long planning horizons which achieve global coverage of the area of interest (non-myopic planning behavior)
- Capability of fast replanning which allows to robustly adapt to external influences such as unknown obstacles

In the course of this thesis, two algorithms have been developed which are highly suitable for the two modules since they meet the imposed requirements. The resulting non-myopic exploration algorithm can be used to explore environmental fields with computationally constrained micro autonomous underwater vehicles. The findings of this work are summarized in the following.

At first, fundamental concepts in probability have been reviewed as a prerequisite for the field belief model. Multivariate Gaussian distributions have been presented to the reader in order to better understand the concepts of Gaussian Processes (GPs). Gaussian Processes are commonly used to describe unknown nonparametric functions through an infinite dimensional Gaussian distribution which is conditioned on noisy observations of the true field. Using a GP as field representation, however, has been concluded to be unsuitable for an online field exploration since a GP does not fulfill the requirements. Therefore, a different belief model is used for **Module I**.

In Chapter 3, Gaussian Markov Random Fields (GMRFs) have been investigated as an approximation of a GP. GMRFs are represented by a finite dimensional multivariate Gaussian distribution which is defined on a grid of fixed dimension. Incorporating the Markov assumption in the GMRF representation, yields a sparsely populated information matrix which enables fast computations of matrix manipulations. A sequential Bayesian update of GMRFs has been derived which allows to update the field belief every time a new measurement is available. The computational complexity of the Bayesian update only depends on the dimension of the grid which results in a constant time needed for the belief update. To justify the use of GMRFs as field representation, it has been compared to other state of the art algorithms in simulation. For that purpose, performance metrics have been defined which are considered important for an underwater field exploration. It has been found that GMRFs are superior to the benchmark algorithms and are, thus, a suitable field belief representation for **Module I**.

A novel approach to the informative path planning problem has been presented in Chapter 4. Based on a literature research, the real-time Rapidly-exploring Random Trees (RT-RRT*), presented in [NaderiRajamakiHamalainen15], have been found as promising method for the path planning module. However, a research gap has been identified since the RT-RRT* algorithm cannot be applied to autonomous exploration missions because the algorithm does not include the concept of information. Therefore, the RT-RRT* algorithm has been extended by an informative framework where each node of the tree is assigned with an

information metric which can be efficiently updated with the knowledge of the GMRF belief. The resulting tree can then be used to plan paths that enable an autonomous exploration of environmental fields. This proposed Information Gathering RT-RRT* (IG-RT-RRT*) algorithm continually rewires a single tree of possible trajectories which is spread in the entire workspace. As a consequence, the informative path planner of **Module II** is able to generate very long paths which can be replanned at a comparably high update rate of 10 Hz. Furthermore, this fast replanning allows to quickly adapt the path to external influences which is needed in the exploration of cluttered environments.

Finally, the resulting path planner has been intensively investigated in simulations in Chapter 5. First, the IG-RT-RRT* algorithm was analyzed regarding its sensitivity to changes in the most influential tuning parameters. The path planner was further compared to two benchmark policies which are given by a random walk policy and a myopic planner. It has been found that the proposed algorithm is superior to the benchmark policies in terms of different performance metrics. Moreover, a case study has been conducted which examined the special cases of source localization and exploration in cluttered environments. In the case study, it has been demonstrated that the path planning module is able to achieve different mission specific goals such as an improved accuracy in areas of high concentration in the source localization scenario. In order to show that the overall exploration algorithm could also be applied in a real world scenario, it was validated in the Gazebo simulation environment which is a sophisticated simulator for robotic systems.

6.2 Outlook

This section briefly presents some future prospects that can be investigated based on the results from this thesis. It has been shown that the proposed planning algorithm is able to explore environmental fields very well in simulations. Using this planning algorithm, the following scenarios could be examined.

Obstacle Information in the GMRF One of the assumptions that was made in the exploration of cluttered environments was that the field belief is independent of the obstacles. This, however, is only true for transmissive obstacles such as fishing nets. An improved GMRF belief could include the obstacle information such that a GMRF edge is deleted whenever it is intersected by an obstacle. The resulting field representation would be more suitable for representing fields with discontinuities due to obstacles. A solution to an obstacle aware GMRF belief has been presented in [MonroyBlancoGonzalez16] where the GMRF edges are adapted to the obstacles which are known in advance. An online update of the GMRF edges

has not been found in the literature and therefore offers possibilities for future work.

Multi Agent Exploration The long planning horizon of the IG-RT-RRT* algorithm enables to early guide the exploration to areas of high uncertainty. Therefore, this algorithm could be directly applied to an exploration with multiple robots if they share a common GMRF belief. This would require a reliable communication link between the robots as they have to share their locations and measurements among each other. Decentralized solutions could be taken into account which tend to be more robust to a loss of communication. In [ViserasXuMerino20], for example, a team of robots cooperatively optimize their paths which are provided by a standard RRT. This cooperative optimization could also be included in a multi agent exploration using the IG-RT-RRT* algorithm which would yield improved paths.

Validation in Experiments Lastly, it is of great interest to validate the performance of the IG-RT-RRT* algorithm in a real world exploration scenario of a cluttered environment. Since the proposed algorithm is especially intended for real-time applications it could be tested on the HippoCampus μAUV to autonomously explore a confined environment. In addition, the provided path could be tracked with a novel controller that was recently developed for agile maneuvering in [Horst21].

Bibliography

[AbbasifardGhahremaniNaderi14] Abbasifard, M.R.; Ghahremani, B.; Naderi, H.: A Survey on Nearest Neighbor Search Methods. International Journal of Computer Applications, Vol. 95, No. 25, pp. 39–52, 2014.

[AsadiEtAl11] Asadi, S.; Pashami, S.; Loutfi, A.; Lilienthal, A.J.: TD Kernel DM+V: Time-Dependent Statistical Gas Distribution Modelling on Simulated Measurements. AIP Conference Proceedings, Vol. 1362, No. 1, pp. 281–282, 2011.

[Besag74] Besag, J.: Spatial Interaction and the Statistical Analysis of Lattice Systems. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 36, No. 2, pp. 192–236, 1974.

[BlancoEtAl13] Blanco, J.L.; Monroy, J.G.; Gonzalez-Jimenez, J.; Lilienthal, A.: A Kalman filter based approach to probabilistic gas distribution mapping. Proceedings of the ACM Symposium on Applied Computing, pp. 217–222, 2013.

[DueckerEtAl19] Duecker, D.A.; Geist, A.R.; Kreuzer, E.; Solowjow, E.: Learning environmental field exploration with computationally constrained underwater robots: Gaussian processes meet stochastic optimal control. Sensors, Vol. 19, No. 9, 2019.

[DueckerEtAl20] Duecker, D.A.; Bauschmann, N.; Hansen, T.; Kreuzer, E.; Seifried, R.: Towards micro robot hydrobatics: Vision-based guidance, navigation, and control for agile underwater vehicles in confined environments. IEEE International Conference on Intelligent Robots and Systems, pp. 1819–1826, 2020.

[ElizabethEtAl12] Elizabeth, F.Q.; Xu, Y.; Choi, J.; Dass, S.; Maiti, T.: Efficient Bayesian Spatial Prediction with Mobile Sensor Networks Using Gaussian Markov Random Fields. American Control Conference, pp. 2171–2176, 2012.

[GalceranCarreras13] Galceran, E.; Carreras, M.: A Survey on Coverage Path Planning for Robotics. Robotics and Autonomous Systems, Vol. 61, No. 12, pp. 1258–1276, 2013.

[GaoShen16] Gao, F.; Shen, S.: Online quadrotor trajectory generation and autonomous navigation on point clouds. SSRR 2016 - International Symposium on Safety, Security and Rescue Robotics, pp. 139–146, 2016.

[Geist18] Geist, A.R.: Combining Gaussian Processes and Stochastic Optimal Control for Non-Myopic Field Exploration with Autonomous Mobile Robots. Master's Thesis MSC-015, Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, 2018.

[Hastedt19] Hastedt, P.: Development and Experimental Validation of a Software-In-The-Loop Simulation Environment for micro Underwater Robots. Project Thesis PRO-034, Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, 2019.

[HollingerSukhatme14] Hollinger, G.A.; Sukhatme, G.S.: Sampling-based robotic information gathering algorithms. International Journal of Robotics Research, Vol. 33, No. 9, pp. 1271–1287, 2014.

[Horst21] Horst, C.H.: Collision free and computational efficient real-time trajectory generation for an Autonomous Underwater Robot. Master's Thesis MSC-040, Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, 2021.

[JonaGianluca13] Jona, G.; Gianluca, L.: Discussing the "big n problem". Statistical Methods & Applications, pp. 97–112, 2013.

[KangEtAl17] Kang, F.; Xu, B.; Li, J.; Zhao, S.: Slope stability evaluation using Gaussian processes with various covariance functions. Applied Soft Computing Journal, Vol. 60, pp. 387–396, 2017.

[KaramanFrazzoli11] Karaman, S.; Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. The International Journal of Robotics Research, Vol. 30, No. 7, pp. 846–894, 2011.

[Kocijan16] Kocijan, J.: Modelling and Control of Dynamic Systems Using Gaussian Process Models. Springer International Publishing, 2016.

[KoenigHoward04] Koenig, N.; Howard, A.: Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. International Conference on Intelligent Robots and Systems (IROS), Vol. 3, pp. 2149–2154, 2004.

[KrauseSinghGuestrin08] Krause, A.; Singh, A.; Guestrin, C.: Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research, Vol. 9, pp. 235–284, 2008.

[KreuzerSolowjow18] Kreuzer, E.; Solowjow, E.: Learning environmental fields with micro underwater vehicles : a path integral - Gaussian Markov random field approach. Autonomous Robots, Vol. 42, No. 4, pp. 761–780, 2018.

[Krige51] Krige, D.G.: A statistical approach to some basic mine valuation problems on the Witwatersrand. Journal of the Southern African Institute of Mining and Metallurgy, Vol. 52, No. 6, pp. 119–139, 1951.

[Lavalle98] Lavalle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. Tech. rep., 1998.

[LilienthalEtAl09] Lilienthal, A.J.; Reggente, M.; Trinca, M.; Blanco, J.L.; Gonzalez, J.: A statistical approach to gas distribution modelling with mobile robots - The Kernel DM+V algorithm. International Conference on Intelligent Robots and Systems, pp. 570–576, 2009.

[LindgrenRueLindström11] Lindgren, F.; Rue, H.; Lindström, J.: An explicit link between gaussian fields and gaussian markov random fields: The stochastic partial differential equation approach. Journal of the Royal Statistical Society. Series B: Statistical Methodology, Vol. 73, No. 4, pp. 423–498, 2011.

[MaLiuSukhatme16] Ma, K.C.; Liu, L.; Sukhatme, G.S.: An information-driven and disturbance-aware planning method for long-term ocean monitoring. IEEE International Conference on Intelligent Robots and Systems, pp. 2102–2108, 2016.

[MalliosEtAl14] Mallios, A.; Ridao, P.; Robas, D.; Carreras, M.; Camilli, R.: Toward Autonomous Exploration in Confined Underwater Environments. Journal of Field Robotics, Vol. 33, No. 1, pp. 1–17, 2014.

[Matérn60] Matérn, B.: Spatial Variation; stochastic models and their application to some problems in forest surveys and other sampling investigations. Ph.D. thesis, Swedish Institute of Experimental Forestry, Stockholm, 1960.

[Mersch19] Mersch, B.: Learning Environmental Fields with an Underwater Robot : Theory and Experiment. Master's Thesis MSC-029, Institute of Mechanics and Ocean Engineering, Hamburg University of Technology, 2019.

[MonroyBlancoGonzalez16] Monroy, J.; Blanco, J.L.; Gonzalez, J.: Time-variant gas distribution mapping with obstacle information. Autonomous Robots, Vol. 40, No. 1, pp. 1–16, 2016.

[MorereMarchantRamos17] Morere, P.; Marchant, R.; Ramos, F.: Sequential Bayesian Optimisation as a POMDP for Environment Monitoring with UAVs. Proceedings - IEEE International Conference on Robotics and Automation, pp. 6381–6388, 2017.

[Murphy13] Murphy, K.P.: Machine learning : a probabilistic perspective. Cambridge, MA, USA: MIT Press, 2013.

[NaderiRajamakiHamalainen15] Naderi, K.; Rajamaki, J.; Hamalainen, P.: RT-RRT*: A real-time path planning algorithm based on RRT*. Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, pp. 113–118, 2015.

[ProcopiucEtAl03] Procopiuc, O.; Agarwal, P.K.; Arge, L.; Vitter, J.S.: Bkd-tree: A dynamic scalable kd-tree. Lecture Notes in Computer Science, Vol. 2750, pp. 46–65, 2003.

[RasmussenWilliams06] Rasmussen, C.E.; Williams, C.K.I.: Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, 2006.

[ReggenteLilienthal10] Reggente, M.; Lilienthal, A.J.: The 3D-kernel DM+V/W algorithm: Using wind information in three dimensional gas distribution modelling with a mobile robot. Proceedings of IEEE Sensors, pp. 999–1004, 2010.

[RueHeld05] Rue, H.; Held, L.: Gaussian Markov Random Fields: Theory And Applications. Chapman & Hall/CRC, 2005.

[StachnissEtAl09] Stachniss, C.; Plagemann, C.; Lilienthal, A.; Burgard, W.: Gas distribution modeling using sparse Gaussian process mixture models. Robotics: Science and Systems, Vol. 4, pp. 310–317, 2009.

[ThrunBurgardFox05] Thrun, S.; Burgard, W.; Fox, D.: Probabilistic robotics. Cambridge, Mass.: MIT Press, 2005.

[ThrunEtAl04] Thrun, S.; Thayer, S.; Whittaker, W.; Baker, C.; Burgard, W.; Ferguson, D.; Hahnel, D.; Montemerlo, M.; Morris, A.; Omohundro, Z.; Reverte, C.; Whittaker, W.: Autonomous exploration and mapping of abandoned mines: Software architecture of an autonomous robotic system. IEEE Robotics and Automation Magazine, Vol. 11, No. 4, pp. 79–91, 2004.

[ValadeEtAl17] Valade, A.; Acco, P.; Grabolosa, P.; Fourniols, J.Y.: A study about kalman filters applied to embedded sensors. Sensors, Vol. 17, No. 12, pp. 1–18, 2017.

[ViserasShutinMerino19] Viseras, A.; Shutin, D.; Merino, L.: Robotic active information gathering for spatial field reconstruction with rapidly-exploring random trees and online learning of gaussian processes. Sensors, Vol. 19, No. 5, 2019.

[ViserasXuMerino20] Viseras, A.; Xu, Z.; Merino, L.: Distributed multi-robot information gathering under spatio-temporal inter-robot constraints. Sensors, Vol. 20, No. 2, pp. 1–25, 2020.

[WatsonDueckerGroves20] Watson, S.; Duecker, D.A.; Groves, K.: Localisation of unmanned underwater vehicles (UUVs) in complex and confined environments: A review. Sensors, Vol. 20, No. 21, pp. 1–35, 2020.

[WiedemannEtAl17] Wiedemann, T.; Shutin, D.; Hernandez, V.; Schaffernicht, E.; Lilienthal, A.J.: Bayesian gas source localization and exploration with a multi-robot system using partial differential equation based modeling. ISOEN 2017 - ISOCS/IEEE International Symposium on Olfaction and Electronic Nose, Proceedings, pp. 10–12, 2017.

Appendix

A.1 Contents Archive

There is a folder **MSC_044_Vahs/** in the archive. The main folder contains the entries

- **MSC_044_Vahs.pdf**: the pdf-file of the thesis MSC-044.
- **Data/**: a folder with all the relevant data, programs, scripts and simulation environments.
- **Latex/**: a folder with the *.tex documents of the thesis MSC-044 written in Latex and all figures (also in *.svg data format if available).
- **Presentation/**: a folder with the relevant data for the presentation including the presentation itself, figures and videos.

A.2 Modifications of the Benchmark Algorithms

A.2.1 Modifications to the Kernel DM+V Algorithm

In [LilienthalEtAl09], the results described in Section 3.3.1 have been used to generate mean and variance maps after a robot has collected measurements, so it has not been applied to online field mapping. For an exploration scenario, however, it is important to update the belief model sequentially since this information affects the decisions of an autonomous robot. In this thesis, a sequential map update is presented which allows for constant computational complexity.

Rather than updating every cell, the grid cell values are stored in matrices with dimension $\mathbb{R}^{n_x \times n_y}$ where n_x and n_y are the discretizations in x and y, respectively. Then, the cell update equations in Eq. (3.41) can be rewritten in recursive form, reading

$$\boldsymbol{\Omega}_t = \boldsymbol{\Omega}_{t-1} + \kappa_2 \left(\boldsymbol{X}, \boldsymbol{x}^{(t)} \right) \tag{A.1}$$

$$\boldsymbol{R}_t = \boldsymbol{R}_{t-1} + \kappa_2 \left(\boldsymbol{X}, \boldsymbol{x}^{(t)} \right) y_t. \tag{A.2}$$

Here, $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n)}]^T$ denotes the set of grid cell center locations and y_t is the most recent measurement taken at location $\boldsymbol{x}^{(t)}$. The confidence map α and the mean map r can be obtained as before by manipulating each individual element of the confidence matrix $\boldsymbol{\alpha}$ and the mean matrix $\boldsymbol{r}$, respectively.

Because of the aforementioned problem that the construction of a variance map has a computational complexity which increases with the number of taken measurements, the uncertainty metric is chosen to be

$$\tilde{\boldsymbol{\Sigma}} = \mathbf{1} - \boldsymbol{\alpha} \in \mathbb{R}^{n_x \times n_y} \tag{A.3}$$

where $\mathbf{1} \in \mathbb{R}^{n_x \times n_y}$ is a matrix filled with ones. This specific choice of metric can be seen as an approximation of the diagonal of a covariance matrix since no covariance between random variables is modeled but only the variance of each random variable. All entries are restricted to $\tilde{\boldsymbol{\Sigma}}_{ij} \in [0, 1)$. This does not match the true variance of the predicted field values but it provides a measure of uncertainty based on how often measurements have been taken at locations of the environmental field. Additionally, the computational complexity of the modified algorithm is constant in the number of measurements and can therefore be applied to online field exploration.

A.2.2 Modifications to the Kalman Filter Based Field Representation

In [BlancoEtAl13], The KF based field representation has been validated with real datasets and has proven its functionality for mapping environmental fields. However, since no kernel functions are used, densely distributed measurements have to be taken in order to get a smooth concentration map. For that reason, a simple but effective modification is proposed in this thesis. The KF representation is extended by a state transition function which allows for extrapolation in unobserved regions using kernel functions. This can be expressed by

$$p\left(\boldsymbol{m}_t|\boldsymbol{m}_{t-1}\right) = \boldsymbol{F}\boldsymbol{m}_{t-1} + \boldsymbol{\varepsilon} \tag{A.4}$$

where $\boldsymbol{F} \in \mathbb{R}^{n\times n}$ is the *state transition matrix* and $\boldsymbol{\varepsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{Q}\right)$ is the Gaussian process noise with covariance matrix $\boldsymbol{Q}$. Note, that the state transition function $p\left(\boldsymbol{m}_t|\boldsymbol{m}_{t-1}\right)$ only depends on the previous state since no control is applied. The proposed state transition matrix is given by

$$\boldsymbol{F} = \begin{bmatrix} \frac{\kappa_2(\boldsymbol{x}_1,\boldsymbol{x}_1)}{\sum_{i=1}^{n_x n_y}\kappa_2(\boldsymbol{x}_1,\boldsymbol{x}_i)} & \frac{\kappa_2(\boldsymbol{x}_1,\boldsymbol{x}_2)}{\sum_{i=1}^{n_x n_y}\kappa_2(\boldsymbol{x}_1,\boldsymbol{x}_i)} & \cdots & \frac{\kappa_2\left(\boldsymbol{x}_1,\boldsymbol{x}_{n_x n_y}\right)}{\sum_{i=1}^{n_x n_y}\kappa_2(\boldsymbol{x}_1,\boldsymbol{x}_i)} \\ \frac{\kappa_2(\boldsymbol{x}_2,\boldsymbol{x}_1)}{\sum_{i=1}^{n_x n_y}\kappa_2(\boldsymbol{x}_2,\boldsymbol{x}_i)} & \frac{\kappa_2(\boldsymbol{x}_2,\boldsymbol{x}_2)}{\sum_{i=1}^{n_x n_y}\kappa_2(\boldsymbol{x}_2,\boldsymbol{x}_i)} & \cdots & \frac{\kappa_2\left(\boldsymbol{x}_2,\boldsymbol{x}_{n_x n_y}\right)}{\sum_{i=1}^{n_x n_y}\kappa_2(\boldsymbol{x}_1,\boldsymbol{x}_i)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\kappa_2\left(\boldsymbol{x}_{n_x n_y},\boldsymbol{x}_1\right)}{\sum_{i=1}^{n_x n_y}\kappa_2\left(\boldsymbol{x}_{n_x n_y},\boldsymbol{x}_i\right)} & \frac{\kappa_2\left(\boldsymbol{x}_{n_x n_y},\boldsymbol{x}_2\right)}{\sum_{i=1}^{n_x n_y}\kappa_2\left(\boldsymbol{x}_{n_x n_y},\boldsymbol{x}_i\right)} & \cdots & \frac{\kappa_2\left(\boldsymbol{x}_{n_x n_y},\boldsymbol{x}_{n_x n_y}\right)}{\sum_{i=1}^{n_x n_y}\kappa_2\left(\boldsymbol{x}_{n_x n_y},\boldsymbol{x}_i\right)} \end{bmatrix} \tag{A.5}$$

with the squared exponential kernel $\kappa_2\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$. Using this process model has the effect that the field concentration at a grid cell is a weighted sum of all other field values based on their distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ and thus allows for advanced prediction in unobserved regions. To incorporate the state transition model in the KF, the predicted belief

$$\bar{\boldsymbol{\mu}}_t = \boldsymbol{F}\boldsymbol{\mu}_{t-1} \tag{A.6}$$

$$\bar{\boldsymbol{\Sigma}}_t = \boldsymbol{F}\boldsymbol{\Sigma}_{t-1}\boldsymbol{F}^T + \boldsymbol{Q} \tag{A.7}$$

has to be updated before the correction step. The predicted or a priori mean $\bar{\boldsymbol{\mu}}_t$ is updated using a deterministic version of the state transition function in Eq. (A.4), with the mean $\boldsymbol{\mu}_{t-1}$ substituted for $\boldsymbol{m}_{t-1}$. Since the covariance matrix $\boldsymbol{Q}$ is positive definite, a prediction step will always increase the predicted covariance $\bar{\boldsymbol{\Sigma}}_t$ which should be used with caution. On the one hand, this offers the opportunity to explore time dependent fields as areas which have been visited before become more uncertain over time which has the consequence that the robot wants to visit that region again. On the other hand, the uncertainty can become arbitrarily large if no observations are made which, however, should not exceed the initial

uncertainty. Therefore, $\boldsymbol{Q}$ can be seen as a tuning parameter to trade-off the uncertainty increase for predictions.

Lastly, to allow for continuous mapping, weighted shape functions are used. In [BlancoEtAl13], the space to be mapped is dicretized in a grid map and it is assumed that measurements that have been taken in a grid cell do not affect neighboring cells. Moreover, within a cell of the grid map, a constant field value is modeled which does not provide a smooth field representation. In this proposed modification, the grid map is replaced by a regular grid $\mathcal{S} = (\mathcal{V}, \mathcal{E})$ consisting of vertices and edges. This grid modification is illustrated in Fig. A.1. Using the proposed grid representation, the overall space can be divided into subdomains $\mathcal{F}^e$. In these subdomains, the field values can be approximated using weighted shape functions which have been covered in section 2.4. As one can see, the marginal distribution of a field value at a location in continuous space can be approximated using the four adjacent vertices. Consequently, the observation

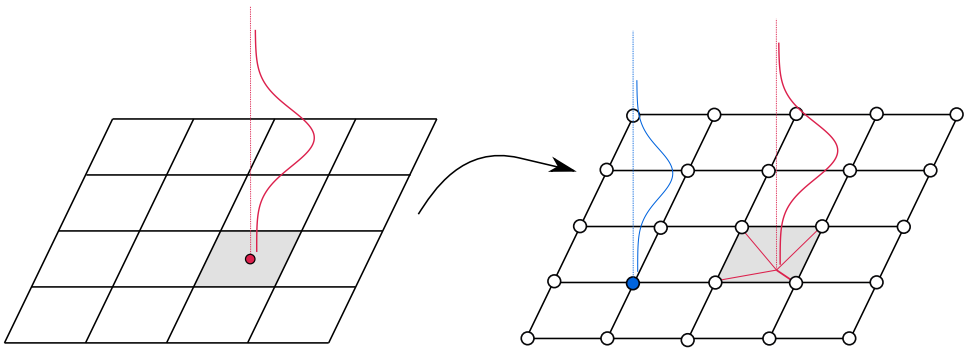


Figure A.1: Illustration of the grid modification to allow for continuous mapping of random variables.

model $\boldsymbol{H}_t$ in Eq. (3.50) and thus the correction step can be improved by including the weighted shape functions

$$\boldsymbol{H}_t = \begin{bmatrix} 0 & \dots & 0 & \phi_1^e & \dots & \phi_4^e & 0 & \dots & 0 \end{bmatrix} \tag{A.8}$$

which have been derived in section 2.4.

Analysis of the Modifications

Subsequently, the proposed modifications are compared to the unmodified Kalman Filter approach. In order to measure the performance of both belief representations, the Root-Mean-Square-Error (RMSE) between the true field and

the mean field belief is used. The RMSE reads

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(\mu_t\left(\boldsymbol{x}_i\right) - f\left(\boldsymbol{x}_i\right)\right)^2}{n}} \tag{A.9}$$

with the total number of random variables n. For an analysis of the belief performance, measurements with simulated noise σ_y are drawn from a true field $f(\boldsymbol{x})$ which is depicted in Fig. A.2. This field is the baseline for analyzing the performance of different field modeling algorithms. In Section 3.4.2, more sophisticated fields are investigated which better reflect the reality.

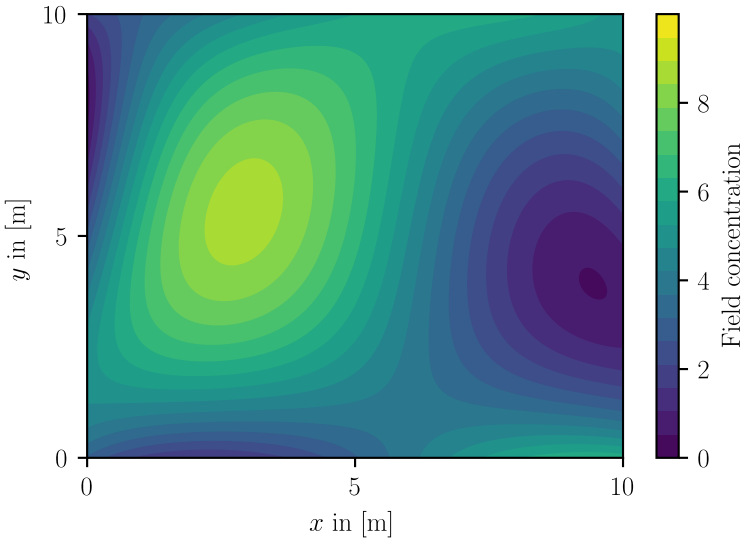


Figure A.2: Smooth concentration field used for generating measurements.

Fig. A.3 and Fig. A.4 show the mean and variance of the belief models after 15 samples have been drawn. The effect of using kernel functions is clearly noticeable. In the unmodified case, a single measurement is only affecting the closest random variable m_i while the surrounding variables remain unchanged which has two reasons. First, since no weighted shape functions are used, a measurement always affects only a single cell. Second, since no extrapolation of field values is used, unobserved locations are unaffected. In the modified case, a measurement also influences random variables in the neighborhood of the measurement location. This happens for the mean concentration value as well as for the predictive variance.

Fig. A.5 shows the RMSE over the number of drawn samples. A significant difference can be seen when the number of samples is low which is due to the extrapolation in unobserved areas. As the number of samples increases, the

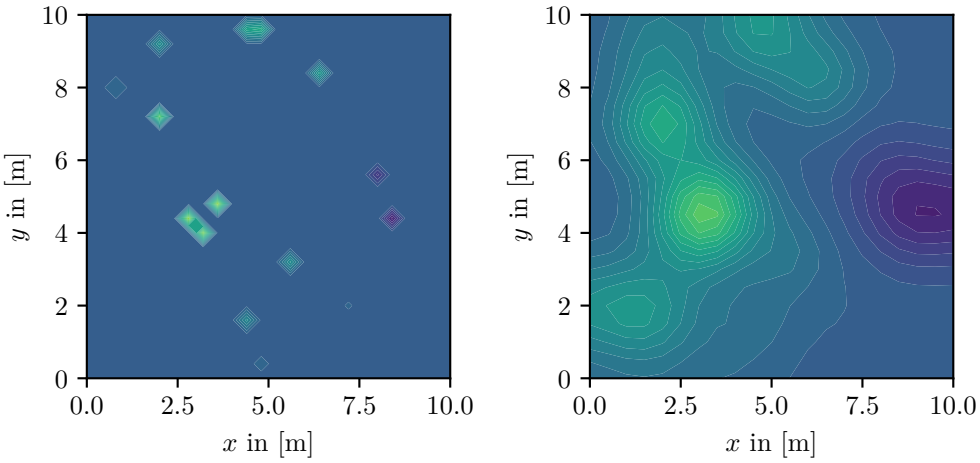


Figure A.3: Mean concentration maps after 15 random samples. Left: Kalman Filter approach from [BlancoEtAl13], Right: Kalman Filter with proposed modifications.

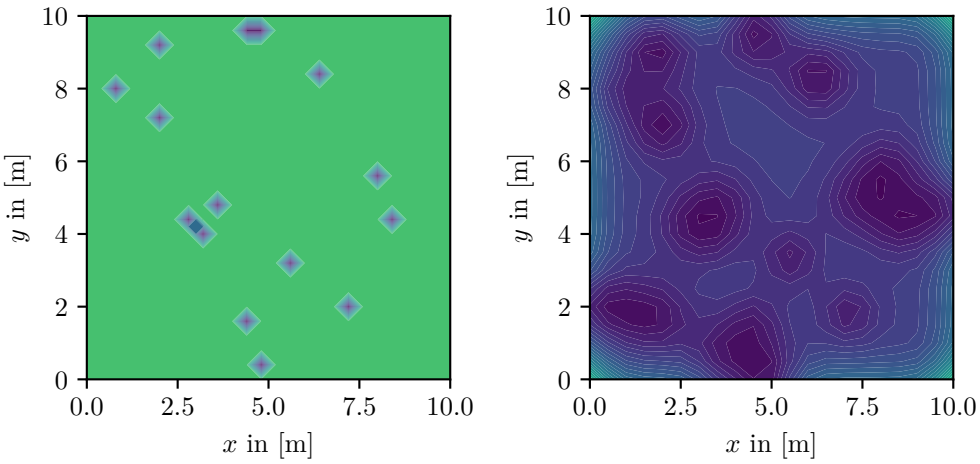


Figure A.4: Maps of the predictive variance using after 15 samples. Left: Unmodified Kalman Filter approach, Right: Kalman Filter with proposed modifications.

RMSE of the unmodified approach gets closer to the RMSE of the proposed approach. After 30 simulations however, the mean RMSE of the modified Kalman Filter is smaller than the unmodified mean RMSE which speaks for a better field representation.

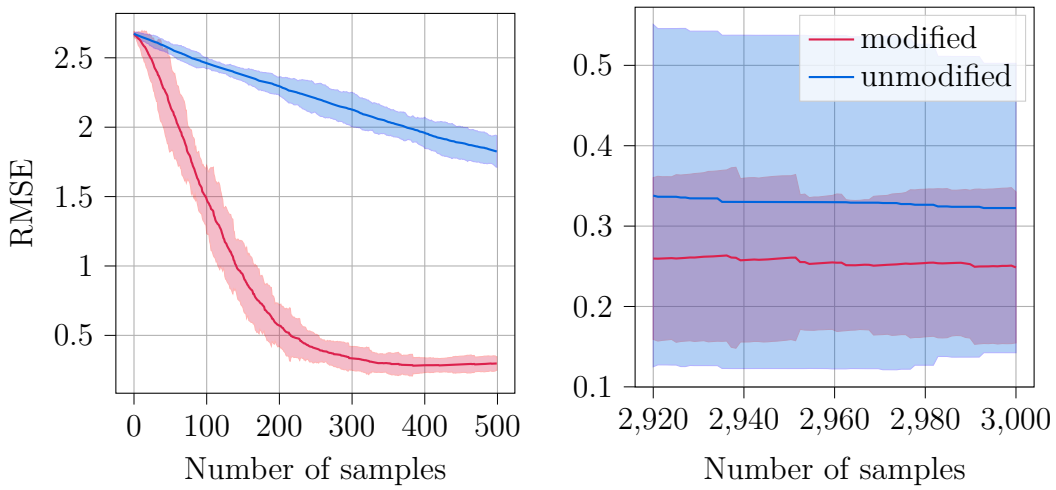


Figure A.5: The mean and 3σ bound of the RMSE over the number of randomly placed measurements.

A.3 Mean Beliefs of Different Fields

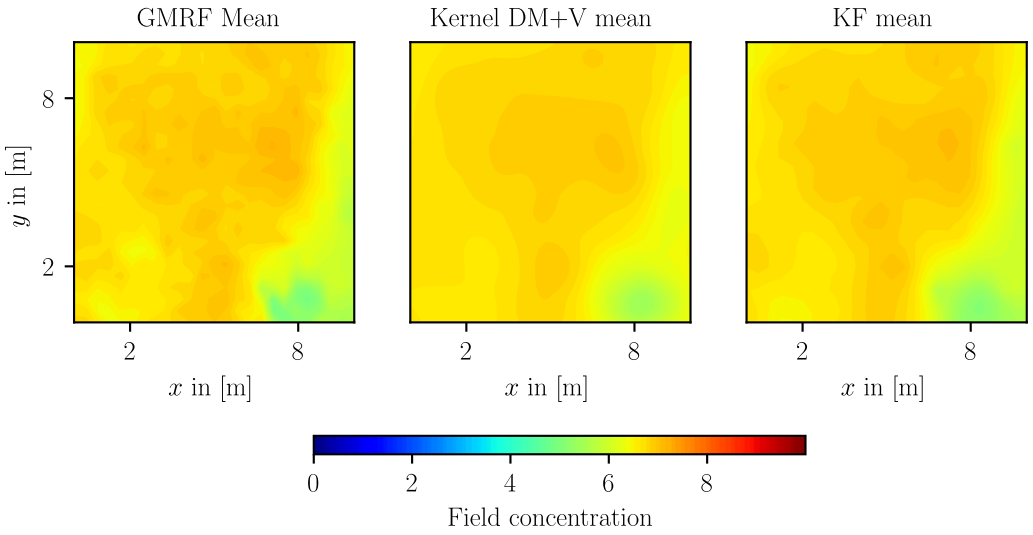


Figure A.6: Mean belief of the different field representations after 500 simulated measurements for **Field II**.

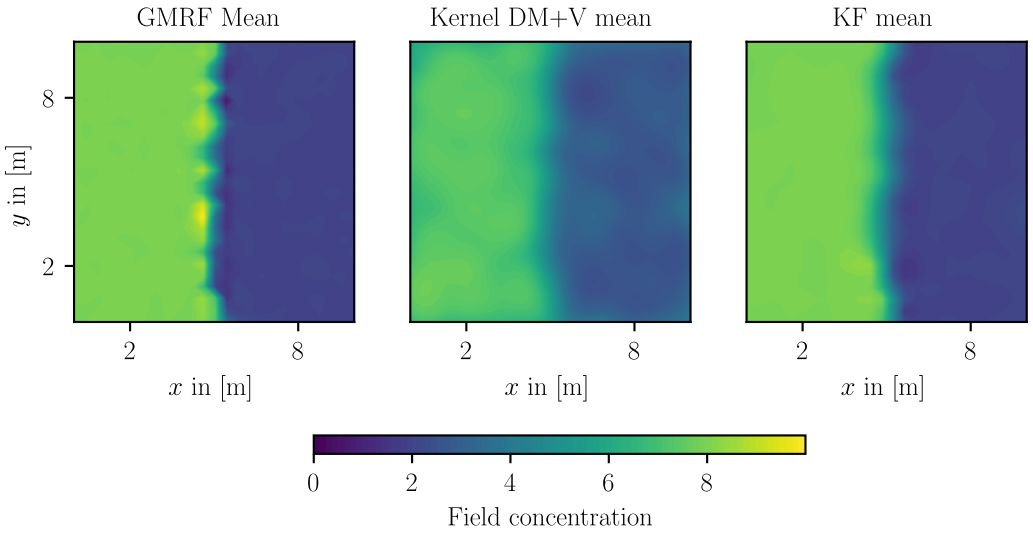


Figure A.7: Mean belief of the different field representations after 500 simulated measurements for **Field III**.

Erklärung

Ich, Matti Vahs (Student des theoretischen Maschinenbaus an der Technischen Universität Hamburg, Matrikelnummer 21599617), versichere, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Unterschrift	Datum