Karlsruhe Institute of Technology

KIT Department of Informatics
Institute for Anthropomatics and Robotics (IAR)
High Performance Humanoid Technologies Lab (H^2T)
Prof. Dr.-Ing. Tamim Asfour

High Performance Humanoid Technologies

Bachelor's Thesis

Multi-fidelity Bayesian Optimization on Riemannian Manifolds

Oswald Paul Zink

16. January 2021

Referee: Prof. Dr.-Ing. Tamim Asfour

Advisors: Dr. Noémie Jaquier
M.Sc. Fabian Reister

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, den 16. Januar 2021

Oswald Paul Zink

Zusammenfassung

Das manuelle Einstellen von Parametern in der Robotik erfordert Expertenwissen auf dem Gebiet und kann viel Zeit in Anspruch nehmen. Ein Beispiel ist ein mobiler Roboter, der den richtigen Griff für einen Türgriff wählen muss: Die optimale Griffpose muss mit wenig Energie erreichbar sein und der Roboter muss danach in der Lage sein die Tür zu öffnen. Um sich die lästige Arbeit der manuellen Auswahl der Greifparameter zu sparen, suchen wir eine automatisierte Möglichkeit diese aufgabenspezifische Parameter zu optimieren. Eine erfolgsversprechende Methode dies zu tun bietet Bayesian Optimization (BO). Hier ist das Ziel mit möglichst wenig Versuchen ein Optimum von einer Funktion zu finden. In der Robotik kann BO angewandt werden indem man eine sogenannte Kostenfunktion manuell entwirft, die von BO daraufhin optimiert wird. Im vorherigen Beispiel könnten Elemente der Zielfunktion die Manipulierbarkeit der Hand nach dem Griff oder die notwendigen Drehmomente für Griff sein.

In vielen Situationen gibt es kostengünstige Approximationen der Zielfunktion. In der Robotik gibt es typischerweise Simulationen, die die reale Welt modellieren. Obwohl die Simulation die reale Welt nur approximiert ist sie kostengünstiger und sicherer auszuwerten. Beispielsweise kann die Simulation beschleunigt werden und Roboterkomponenten können nicht beschädigt werden. Das Konzept des Multi-fidelity BO (MF-BO) verwendet weitere kostengünstige Informationsquellen, wie die Simulation in der Robotik, und nimmt sie in das BO Verfahren auf. Dabei nutzt das Verfahren die ungenaue kostengünstige Informationsquelle um die Zielfunktion grob zu erkunden.

Desweiteren liegen viele genutze mathematische Strukturen in der Robotik nicht im euklidischen Raum, sondern in Riemannschen Mannigfaltigkeiten. Ein Beispiel dafür sind Regelverstärkungsmatrizen, welche im Raum der symmetrischen positiven Matrizen liegen. Das sogenannte

geometry-aware BO (GaBO) nutzt diese Information der Datenstruktur aus um effizienter zu optimieren.

BO ist schon seit einer langer Zeit in der wissenschaftlichen Literatur zu finden [Močkus, 1975] und wird auch im Feld der Robotik angewandt [Marco et al., 2016], [Doerr et al., 2017], [Jaquier, 2020]. GaBO steht noch auf jungen Füßen in der Wissenschaft. Dennoch gibt es schon vielversprechende Verbesserungen gegenüber klassischen BO Varianten [Jaquier et al., 2019]. Obwohl MF-BO seit kurzem im Fokus der Wissenschaft steht [Wu et al., 2020], [Marco et al., 2017], [Kandasamy et al., 2016], [Song et al., 2019] gibt es sehr viele verschiedene Herangehensweisen. Daher ist auf diesem Gebiet noch weitere Forschung nötig. Forschung, die beide Felder MF-BO und GaBO zusammen betrachtet sind uns zum Zeitpunkt der Arbeit unbekannt.

Im Rahmen dieser Arbeit haben wir ein Optimierungsverfahren entwickelt, das die Geometrie berücksichtigt und gleichzeitig die Simulation als zweite Informationsquelle hat. Wir stellen unseren Algorithmus MF-GaBO vor und testen ihn auf der Currin- und Borehole-Testfunktion. Zusätzlich haben wir dieses Verfahren an dem Roboter ARMAR-6 evaluiert, indem wir die Greifposition und -orientierung des Roboters ARMAR-6 zum Greifen eines Geschirrspülergriffes optimiert.

Abstract

Manual tuning of robotics parameters usually requires the expertise of experts in the field and can be time consuming. For example, consider the case of a mobile robot that has to select the right grasp pose to grasp a door handle: The optimal grasp pose must be reachable by the robot with little energy and allow it to then easily open the door. In order to avoid the burden of manually choosing this grasp pose, we are interested in methods that can automatically optimize these task-specific parameters of interest. A promising approach to do so is Bayesian Optimization (BO). The goal of Bayesian Optimization is to optimize, i.e., find an optimum of a function with as little trials as possible. In robotics BO can be used by providing an objective function that is being optimized which is usually hand crafted. In the previous example the objective function could consist of several criteria, e.g., the manipulability of the hand after grasping or the amount of torques needed to grasp.

In some situations cheap approximations of the target function are obtainable. In robotics, simulators typically provide an approximation of the real world. Although the simulation is just an approximation, the evaluation is more convenient, e.g., the simulation can be sped up, and run in parallel and in the simulation the robot can not be damaged. Multi-fidelity BO (MF-BO) tries to incorporate information sources of lower fidelity in order to find the optimum even faster in terms of cost by leveraging the low cost of the lower fidelities to explore the objective function cheaply.

In robotics several commonly used data structures do not lie in Euclidean spaces but rather in Riemannian Manifolds. An example of this are controller gain matrices which lie in the space of symmetric positive definite matrices. In this context, geometry-aware BO (GaBO) incorporates domain knowledge of the optimized parameters to further improve BO.

BO has been scientifically studied for a long time [Močkus, 1975] and has re-emerged in robotics [Marco et al., 2016], [Doerr et al., 2017], [Jaquier, 2020]. GaBO has only recently emerged as one of the offsprings of BO but has shown improvement over traditional non geometry-aware approaches [Jaquier et al., 2019]. Although MF-BO recently gained interest among the research community [Marco et al., 2017], [Wu et al., 2020], [Kandasamy et al., 2016], [Song et al., 2019] most the approaches greatly vary and further research is necessary.

In this thesis we propose a multi-fidelity geometry-aware BO (MF GaBO) approach that incorporates both domain knowledge and approximations of the target function as lower fidelities. We propose an algorithm for MF-GaBO and test our approach on the Currin and Borehole benchmark functions. Furthermore, we evaluate our approach on the humanoid robot ARMAR-6 by optimizing the pose of a contact-rich manipulation task.

Table of Contents

Chapter 1

Introduction

When optimizing very costly experiments, such as drilling holes to find oil then it is mandatory to try to make as few as possible experiments to find an optimum. In this example the optimum is an oil source. Bayesian Optimization attempts to do exactly this. It models the objective function via a surrogate model. Most of the time a Gaussian process is chosen as surrogate model. More detailed information about BO can be found in section 2.2. In some settings there is an approximation of the objective function which is cheaper to evaluate but not as accurate as the true objective function. With this concept the idea of doing as few experiments as possible can be further pursued by doing less experiments on the highest fidelity and more on lower cheaper fidelities. Here cheap could mean anything, e.g., money, computation time, etc.. In robotics, oftentimes there exists a simulation which models the robot and the world with a certain accuracy but is a lot cheaper to evaluate than a experiment on the real robot. Incorporating all fidelities into BO is called Multi Fidelity Bayesion Optimization or Multi Fidelity Bandit Optimization. One more recent adaptation of BO is geometry-aware Bayesian Optimization, first introduced by Jaquier et al. [2019]. In some applications the search space is not the Euclidean space but some other Riemannian Manifold. For example in robotics, orientations are often described by unit quaternions which lie on the $\mathcal{S}^3$ space. In these spaces Euclidean metrics are no longer valid. Therefore, to properly represent the problem the kernel and the optimization of the acquisition function have to be adjusted. In this thesis we combine

the multi-fidelity and the geometry-aware approach and test the methodology on a humanoid robot: ARMAR-6.

BO has been scientifically studied for a long time [Močkus, 1975] and has re-emerged in different fields such as robotics [Marco et al., 2016], [Doerr et al., 2017], [Jaquier, 2020] and machine learning [Wu et al., 2020]. The idea of incorporating lower fidelity information sources into BO is called multi-fidelity BO. This field is already being studied and has several different approaches, e.g., different applications require either a discrete or a continuous fidelity space. Just like there exist many acquisition functions in BO literature [Močkus, 1975], [Srinivas et al., 2009], [Hennig and Schuler, 2012], there are also many different approaches on how to incorporate all the fidelities into a multi-fidelity approach Wu et al. [2020], [Kandasamy et al., 2016], [Marco et al., 2017], [Song et al., 2019]. Geometry-aware BO has only recently emerged as one of the offsprings of BO but has shown improvement over traditional non geometry-aware approaches [Jaquier et al., 2019] and has several applications in robotics such as optimizing orientation, manipulability or path finding problems [Jaquier et al., 2021].

The field of geometry-aware BO and multi-fidelity BO has previously only been studied separately. Thus, in this thesis we introduce a novel multi-fidelity geometry-aware BO algorithm (MF-GaBO) that combines both fields. As proof of concept we test MF-GaBO on two benchmark functions and on a contact-rich manipulation task on a humanoid robot: ARMAR-6.

First, in chapter 2 we introduce the necessary theoretical background on Riemannian manifolds, Bayesian Optimization, Gaussian Processes and multi-fidelity Bayesian Optimization. Then, in chapter 3 we discuss the state of the art of MF-BO and GaBO. Afterwards, in chapter 4 we introduce the proposed MF-GaBO algorithm. In chapter 5 we discuss the implementation details of software used to conduct the experiments on the humanoid robot ARMAR-6. Finally, in chapter 6 we discuss the conducted experiments on the benchmark functions, on the humanoid robot ARMAR-6 and on the multi-fidelity setup with ARMAR-6 and a simulation. Additionally in this chapter, we also interpret the results of the experiments and conclude that the simulation in its current state does not benefit to our MF-GaBO setup. Lastly, in chapter 7 we summarize our findings and discuss about possible future research.

Chapter 2

Fundamentals

2.1 Riemannian Manifolds

Intuitively, a Riemannian manifold $\mathcal{M}$ is a smooth, differentiable topological space in which each point $\boldsymbol{x} \in \mathcal{M}$ locally resembles Euclidean space [Do Carmo and Flaherty Francis, 1992]. In robotics some of the commonly used data does not lie in Euclidean space. Examples are the unit quaternion used to represent orientations and symmetric positive definite (SPD) matrices used to represent stiffness and inertia matrices. These spaces can be adequately handled with Riemannian manifolds. The unit quaternion belongs to the sphere $\mathcal{S}^3$ space where the sphere is defined as $S^n = \{\boldsymbol{x} \in \mathbb{R}^{n+1} : ||\boldsymbol{x}|| = 1\}$. Intuitively, this is can be explained because the unit quaternion is normalized and has 4 parameters. In this thesis we use the $\mathcal{S}^3$ space to represent orientation when grasping a dishwasher handle.

Geodesics Instead of Euclidean distances on Riemannian manifolds geodesics describe the shortest path between two points on a manifold. Figure 2.1 shows the Euclidean distance and geodesic of two points on a sphere manifold. Instead of straight lines, as in Euclidean space, the distance is measured as minimum over all possible curves between two points on the manifold. The geodesic distance measurement for the sphere manifold $\mathcal{S}^d$ is defined as:

$$d_{\mathcal{S}^d}(\boldsymbol{x}, \boldsymbol{y}) = \arccos(\boldsymbol{x}^\intercal \boldsymbol{y}) \tag{2.1}$$

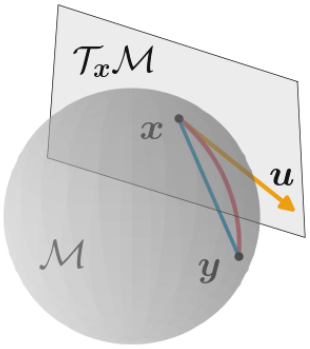


Figure 2.1: Two points x and y on a sphere manifold $\mathcal{M}$. The Euclidean distance is shown in blue. The geodesic distance is shown in red. The tangent space $\mathcal{T}_x\mathcal{M}$ of the point x is shown as transparent gray area. Figure taken from [Jaquier et al., 2019].

Tangent spaces Classical Euclidean operations can not be done in Riemannian Manifolds because the space is not closed under addition and scalar product [Do Carmo and Flaherty Francis, 1992]. Instead, these operations are performed on points $\boldsymbol{x} \in \mathcal{M}$ are performed in the corresponding tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ which is Euclidean. The tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ is formed by all tangent vectors to all 1-dimensional curves on $\mathcal{M}$ that pass trough $\boldsymbol{x}$. By definition the origin of the tangent space coincides with $\boldsymbol{x}$. To transfer points between the manifold $\mathcal{M}$ and the tangent space $\mathcal{T}_{\boldsymbol{x}}$ we make use of this the exponential map and logarithmic map. The exponential map

$$\mathrm{Exp}_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}, \quad \boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) \quad \text{with} \quad \boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{M} \tag{2.2}$$

maps a point $\boldsymbol{u}$ on the tangent space of $\boldsymbol{x}$ to a point $\boldsymbol{y}$ on the manifold. The logarithmic map

$$\mathrm{Log}_{\boldsymbol{x}} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}, \quad \boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) \quad \text{with} \quad \boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{M} \tag{2.3}$$

is the inverse of the exponential map. In Figure 2.1 the point $\boldsymbol{u}$ is the result of using the logarithmic map on the point $\boldsymbol{y}$: $\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y})$. Analogically, the point $\boldsymbol{y}$ is the result of using the exponential map on the point $\boldsymbol{y}$: $\boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$

Product of Riemannian Manifolds The Cartesian product of two Riemannian manifolds $\mathcal{M}_1 \times \mathcal{M}_2$ is a Riemannian manifold. Specifically, it is the set of pairs $(\boldsymbol{x_1}, \boldsymbol{x_2})$ with $\boldsymbol{x_1} \in \mathcal{M}_1, \quad \boldsymbol{x_2} \in \mathcal{M}_2$. On a product of manifolds the squared distance is the squared distance on each individual manifolds:

$$d^2_{\mathcal{M}_1 \times \mathcal{M}_2}((\boldsymbol{x_1}, \boldsymbol{x_2}), (\boldsymbol{y_1}, \boldsymbol{y_2})) = d^2_{\mathcal{M}_1}(\boldsymbol{x_1}, \boldsymbol{y_1}) + d^2_{\mathcal{M}_2}(\boldsymbol{x_2}, \boldsymbol{y_2})$$

with $\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{M}_1$ and $\boldsymbol{x_2}, \boldsymbol{y_2} \in \mathcal{M}_2$ [Lee, 2018].

The logarithmic and exponential map are calculated by applying the maps on the individual manifolds [Lee, 2018], e.g.,

$$\mathrm{Log}_{\boldsymbol{x_1}, \boldsymbol{x_2}}((\boldsymbol{y_1}, \boldsymbol{y_2})) = (\mathrm{Log}_{\boldsymbol{x_1}}(\boldsymbol{y_1}), \mathrm{Log}_{\boldsymbol{x_2}}(\boldsymbol{y_2}))$$

The needed adjustments for Bayesian Optimization to be geometry-aware and thus considering Riemannian Manifolds are discussed in subsection 2.3.1.

2.2 Bayesian Optimization

The goal of Bayesian Optimization (BO) is to find a global optimum of a black box function in a minimal number of steps [Frazier, 2018], [Močkus, 1975], [Shahriari et al., 2015]. A black box function is a function for which no prior information is available, but that can be observed at each point. Therefore, a so-called surrogate model is used to model the current belief about the underlying function. This approximation of the black box objective function is then updated with the information gained in each iteration of BO. Most often a Gaussian Process is used as a surrogate model. Bayesian Optimization uses a so-called acquisition function in combination with the surrogate model to calculate the next point to query. This is the key of Bayesian Optimization: it determines the point that will be most beneficial in finding the global maximum. Acquisition functions make a trade-off between exploitation and exploration. When exploiting, the acquisition function queries near points $\boldsymbol{x}$ at which the surrogate model believes good values are likely. When exploring, the acquisition function tries to query uncertain areas of the surrogate model that due to the uncertainty might yield good values.

Gaussian processes

For a given set of training data $X = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}, \quad \boldsymbol{x_i} \in \mathcal{X} \qquad \forall i = 1, ..., n$ there exist infinitely many functions that fit the data. A Gaussian Process (GP) allows the assignment of probabilities to these functions:

$$f(\boldsymbol{x}) \sim \mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}))$$

with a mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

A stochastic process is a collection of random variables indexed by space or time. Formally, a Gaussian Process is a stochastic process that is defined so that this collection of random variables (which also can be infinite) is a multivariate normal distribution (MVN). When we try to estimate a function in space, e.g. $\mathbb{R}$, then the GP is indexed by space. A MVN is defined by its mean μ and covariance matrix $\boldsymbol{K}$. When we use a GP to model an underlying black function we assume that the function is smooth. Therefore, inputs that lie close together in the input space are expected to have similar values in the output space. To model this assumption we use kernel functions k that determine the similarity between two input points $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ to calculate the covariance matrix.

The goal of a GP is to learn the underlying distribution of the functions $f(\boldsymbol{x})$ at any point of space. Therefore, we need to be able to evaluate the current belief of the GP at any point $\boldsymbol{x}$ of the function $f(\boldsymbol{x})$. To do so, the idea is to have a joint distribution of the training points X and the points that we want to evaluate X_* with $f(X_*) = Y_*$ which is then conditioned on X using Bayesian inference. The essential idea of Bayesian inference is to update the hypothesis as new information is gained. In the context of GPs the information is gained by having more training data X [Rasmussen, 2003b]. Thus, we are interested in the conditional probability $P_{X_*|X}$. Without conditioning the training data in the context of Bayesian optimization this is called the prior distribution:

$$P\begin{pmatrix} Y \\ Y_* \end{pmatrix} = \mathcal{N}(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} \boldsymbol{K} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{pmatrix}) \tag{2.4}$$

with $\boldsymbol{K} = k(X, X)$, $\boldsymbol{K}_{**} = k(X_*, X_*)$ and $\boldsymbol{K}_* = k(X, X_*)$. Usually, the mean μ is chosen as 0. We incorporate prior knowledge into BO by choosing how the covariance matrix K of the GP used as surrogate model is calculated. Namely, we choose the kernel function k which is used to calculate the covariance matrix $\boldsymbol{K}_{ij} = k(x_i, x_j)$. With different kernels the prior changes, e.g., with a periodic kernel a periodic function is estimated. This is because the periodic kernel not only seems points close in space similar but does so periodically.

Given a set of training data $\{X, Y\}$, we can use Bayesian inference to condition the joint prior distribution Equation 2.4 and determine the output Y_* corresponding to a new input X_*:

$$P_{Y_*|Y,X,X_*} \sim \mathcal{N}(\mu_{\text{new}}, \boldsymbol{K}_{\text{new}})$$
$$\mu_{\text{new}} = \mu_{X_*} + \boldsymbol{K}_*^T \boldsymbol{K}^{-1}(X - \mu_X)$$
$$\boldsymbol{K}_{\text{new}} = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \boldsymbol{K}^{-1} \boldsymbol{K}_*$$

This posterior distribution evaluated at any point x_* represents the model prediction and uncertainty of the function $f(\boldsymbol{x_*})$ at $\boldsymbol{x_*}$.

Kernels As mentioned earlier the kernel function k determines how similar different points in space are. Some of the most commonly used kernels are the radial basis function (RBF) Kernel and the Matérn Kernel. They are defined as Rasmussen [2003a], Duvenaud [2014]:

$$k_{\text{Matérn}}(\boldsymbol{x_1}, \boldsymbol{x_2}) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu} \frac{d}{\rho} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{d}{\rho} \right)$$

with Γ being the Gamma function, K_ν being the modified Bessel function and d being the Euclidean distance between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ scaled by the lengthscale parameter Θ, i.e.,

$$d = (\boldsymbol{x_1} - \boldsymbol{x_2})^\intercal \Theta^{-2} (\boldsymbol{x_1} - \boldsymbol{x_2}) \tag{2.5}$$

and ν being the smoothness parameter (smaller values are less smooth). In this thesis we use $\nu = \frac{5}{2}$. There are several other kernel functions such as the periodic kernel used to model periodic functions. As we can see the kernel functions depend on the distance between two points x_i and x_j. Therefore, to design geometry-aware kernels we have to use the geodesic distance instead of the Euclidean distance which will be discussed in subsection 2.3.1.

Acquisition functions

Acquisition functions are used to determine the next point to query. The goal is to find the optimum in as few queries as possible, therefore the acquisition function is the cornerstone of BO. All acquisition functions make a trade off between exploration and exploitation. When exploring, the acquisition function chooses a point where the surrogate model is uncertain about. When exploiting, the acquisition function chooses a point close to a point where the surrogate model expects a good value.

There are several commonly used acquisition functions: Upper Confidence Bound (UCB) [Srinivas et al., 2009], Expected Improvement (EI) [Močkus, 1975] and Entropy Search (ES) [Hennig and Schuler, 2012]. In this thesis we make use of the UCB acquisition function. UCB uses the sum of the mean and the standard deviation of the posterior to determine how desirable a point is, i.e.,

$$U(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sqrt{\beta} \cdot \sigma(\boldsymbol{x}) \tag{2.6}$$

In UCB the exploration is represented by $\sqrt{(}\beta) \cdot \sigma(\boldsymbol{x})$. When the uncertainty $\sigma^2(\boldsymbol{x})$ is high then this term also grows. On the other hand, when exploiting the acquisition function tries to choose near a point that already promises good objective function values. In UCB this is represented by $\mu(\boldsymbol{x})$. When the mean of the surrogate model is high then the acquisition function tries to further exploit this area.

2.2.1 Multi-fidelity Bayesian Optimization

The idea of multi-fidelity BO (MFBO) is to incorporate several fidelities of the problem into BO. In this context a fidelity describes how accurate an information source is. A two fidelity example is a robot in the real world and a simulation in which the robot is modeled. In this example the simulation is the lower fidelity as it approximates the real world. Most of the time the lower fidelities are cheaper to evaluate. In the previous example the simulation might cost less expense than setting up a real robot. Ideally, with MF-BO the cheap lower fidelity can be used to explore the problem in order to find the optimum in the higher fidelity faster with less iterations and therefore saving expense. Let there be a budget Λ to spend and each fidelity $f^{(1)} \dots f^{(n)}$ has a cost $\lambda^{(1)} < \dots < \lambda^{(n)}$. Again, the goal is to reduce the amount of spent budget Λ to find the optimum. Ideally, the information gained from the lower fidelities should be incorporated in the BO setup so that less queries are needed in the original fidelity.

One idea to implement this idea is to use one surrogate model for each fidelity [Kandasamy et al., 2016]. At each iteration all fidelities are considered and depending on how certain the corresponding surrogate models are, the next fidelity to query is chosen.

There are several other approaches to MF-BO which are discussed in detail in chapter 3. In section 6.1 we show that in theoretical benchmark the multi-fidelity setup performs better than the single-fidelity alternative. In this thesis, we introduce a geometry-aware MF-BO algorithm

(MF-GaBO) with the goal of optimizing a contact-rich manipulation task. The goal was to use MF-GaBO with two fidelities: the humanoid robot ARMAR-6 as high fidelity and a simulation of the robot as lower fidelity.

2.3 Bayesian Optimization on Riemannian Manifolds

In some applications the search space is not the Euclidean space but some other Riemannian manifold. For example in robotics, orientations are often described as unit quaternions which lie on the sphere manifold $\mathcal{S}^d$. In these spaces Euclidean metrics are no longer valid. Therefore, to properly represent the problem the kernel and the optimization of the acquisition function have to be adjusted. By doing so, the problem turns from a constrained problem in Euclidean space into a unconstrained problem in the Riemannian manifold. In the unit quaternion example if optimized in Euclidean space the optimization has to be constrained on values that have norm 1. On the other hand, when optimizing the problem on the sphere manifold all points intrinsically lie on the surface on the sphere and are thus already normalized. Therefore the problem is then unconstrained. The advantage of having an unconstrained problem is faster and better convergence.

2.3.1 Kernels

As shown in section 2.2 a fundamental part of every kernel is the distance measurement between two points. This distance measure traditionally is the Euclidean distance. This is correct for Euclidean spaces. However, because we optimize on non Euclidean Riemannian manifolds, e.g., over the orientation in our experiments we modify the kernel to use the geometry-aware distance measures for $\mathcal{S}^3$.

Intuitively, one would replace the Euclidean distance in the kernels by the geodesic distance. For example, to make the Matérn kernel Equation 2.5 geometry-aware the Euclidean distance d would be replaced by the geodesic distance $d_{\mathcal{M}}$:

$$k_{\text{Matérn}}(\boldsymbol{x_1}, \boldsymbol{x_2}) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\sqrt{2\nu}\frac{d_{\mathcal{M}}}{\rho}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{d_{\mathcal{M}}}{\rho}\right) \tag{2.7}$$

Because the naive formulation in Equation 2.7 is only valid, e.g., positive definite (PD) for all parameter values, if the manifold is isometric to an Euclidean space [Jaquier et al., 2019],

Borovitskiy et al. [2020] introduces valid so-calledMatérn kernels using spectral theory of the Laplace-Beltrami operator which we use in this thesis:

$$k_{\text{Matérn}}(\boldsymbol{x_1}, \boldsymbol{x_2}) = \frac{\sigma^2}{C_\nu} \sum_{n=0}^{\infty} \left(\frac{2\nu}{\kappa^2} + n(n+d+1) \right)^{-(\nu+\frac{d}{2})} c_{n,d} C_n^{(d-1)/2}(\cos(d_{\mathcal{M}}(\boldsymbol{x_1}, \boldsymbol{x_2}))) \quad (2.8)$$

with $C_n^{(d-1)/2}$ being the Gegenbauer polynomials and the constant $c_{n,d}$ being a constant defined as $c_{n,d} = \frac{d_n \Gamma((d+1)/2)}{2\pi^{d+1)/2} C_n^{(d-1/2)}(1)}$. In this thesis we use the Equation 2.8 with the parameter set as $\nu = 2.5$.

Acquisition functions

When doing Bayesian optimization the acquisition function is maximized to find the next point to query. Conventionally, constrained optimization is performed to optimize on e.g. $\mathcal{S}^d$. An alternative to this are optimization methods for Riemannian manifolds in which the constraints are implicitly taken care of by optimizing on the manifold, therefore making it an unconstrained optimization [Jaquier et al., 2019]. Examples of this are the conjugate gradient (CG) algorithm on Riemannian manifolds [Absil et al., 2008] and Trust Regions on Riemannian Manifolds [Absil et al., 2008] which we use in this thesis. Additionally, in this thesis we use the constrained Trust regions on Riemannian manifolds presented in [Jaquier et al., 2019] to optimize the acquisition function when additional constraints on the manifold must be taken into account, e.g., when restricting the range of available orientations.

Chapter 3

Related Work

3.1 Bayesian Optimization

Bayesian Optimization has been used since the 1960s for designing engineering system and more recently since 2012 for optimizing deep neural networks [Frazier, 2018] and for robot tasks [Marco et al., 2016]. BO is also used in robotics because it aims to be information efficient, e.g., reducing the amount of robot experiments. Additionally, BO allows the automation of the traditionally manual fine tuning of parameters. In robotics BO is already being researched. Marco et al. [2016] use BO to optimize the balancing of poles of different lengths. Yuan et al. [2019] use a modified BO to optimize parameters for whole-body control.

Multi-fidelity approaches Papers on multi-fidelity BO started emerging in 2004 [Frazier, 2018]. Research is not only applied on theoretical benchmark functions [Kandasamy et al., 2016], but also on different fields such as optimizing hyperparameters of deep neural networks [Wu et al., 2020] and optimizing control policies in robotics [Marco et al., 2017]. There are different approaches to implementing MF BO. The approaches differ how each fidelity is modeled, whether the fidelity space is continuous or discrete and most importantly how the information from lower fidelities is transferred to the goal fidelity.

Marco et al. [2017] use a specific kernel structure to incorporate two information sources. As lower fidelity they use a simulation and as high fidelity a robot experiment. They extend each training input $\boldsymbol{x}$ with an additional binary parameter θ that represents whether the experiment or the simulation is being queried. Thus, the acquisition function directly determines which fidelity is queried next. Therefore, they use one modified kernel to represent the two fidelities. With this approach Marco et al. [2017] optimize control policies of a cart-pole system.

In contrast, Kandasamy et al. [2016] uses one GP for each fidelity. In order to share the information gained from lower fidelities, the acquisition function proposed by Kandasamy et al. [2016] checks every fidelity at each iteration. At each iteration first the next point to query $\boldsymbol{x}_{\text{new}}$ is being determined by using an acquisition function that considers every fidelity. Then the fidelity that will be queried next is determined by checking the confidence of each surrogate model for each fidelity at the determined point $\boldsymbol{x}_{\text{new}}$.

Wu et al. [2020] use MF BO to optimize hyperparameters for deep neural networks. The authors see fidelities as multiple continuous dimensions, e.g., iterations, training data and validation data and propose a trace-aware approach that saves the information gained for all lower fidelities when training a deep neural network. For example, if a neural network is trained with x iterations then the information for the lower fidelities with $0, ..., x-1$ iterations is also gained. Because they also extend the input domain like Marco et al. [2017] they use a single GP to model the fidelities. Wu et al. [2020] use a novel modified version of knowledge-gradient as acquisition function.

Song et al. [2019] use a multi-output joint GP to model n fidelities. The authors propose an algorithm that maximizes the mutual information gained across all fidelities. To evaluate their work, the authors demonstrate the effectiveness of their approach in theoretical benchmarks, e.g., the Currin and Borehole function and in real world experiments, e.g., experimental design for material science.

Geometry-aware Bayesian Optimization Geometry-aware Bayesian Optimization (GaBO) has only recently emerged as an extension of BO by [Jaquier, 2020]. In [Jaquier et al., 2019] the authors use GaBO to optimize the orientation and impedance parameters on a 7-DOF simulated robot by using kernels that use geometry-aware distances instead of Euclidean distances. Later, Borovitskiy et al. [2020] shows that the kernels used in Jaquier et al. [2019] are not always valid and and introduces valid Matérn kernels using spectral theory of the Laplace-Beltrami operator. Jaquier et al. [2021] then use these novel kernels to optimize orientation, manipulability and a path planning problem in a robot simulation.

Chapter 4

Concept

4.1 General idea

The main idea of this thesis is to design a multi-fidelity geometry-aware BO setup in order to incorporate a robotic simulator as an additional information source and the underlying geometry into traditional BO. To do so, we extend the currently existing MF BO approach of Kandasamy et al. [2016] to be geometry-aware. We chose Kandasamy et al. [2016]'s approach because it uses a modified version of the Upper Confidence Bound (UCB) acquisition function. Other MF approaches use Entropy Search (ES) as acquisition function which requires sampling which is not straightforward to extend on manifolds. First we test MF-GaBO on benchmark functions (see section 6.1). Afterwards, we evaluate the MF-GaBO setup on a contact-rich robot manipulation task in which we optimize the handpose of grasping of a dishwasher handle.

4.2 Multi-Fidelity Geometry-aware Bayesian optimization

In our multi-fidelity setup we use the approach of Kandasamy et al. [2016]. Although, the setup proposed by Kandasamy et al. [2016] is formulated for N fidelities, we only use 2 fidelities for all conducted experiments. Namely the lower fidelity is a simulation (sim) and the higher

fidelity is a real robot experiment (exp). Kandasamy et al. [2016]'s approach uses one GP for each fidelity and uses an acquisition function that considers every fidelity to find the point to query. Only after deciding the point to query Kandasamy's approach decides which fidelity to query based on how certain the GPs of the fidelities are.

Our contribution to Kandasamy et al. [2016]'s approach is using geometry-aware kernels and a geometry-aware optimization method. Instead of a classical Euclidean kernel we use the geometry-aware version introduced by Jaquier et al. [2021]. Furthermore, we use a Trust Regions on Riemannian manifolds [Absil et al., 2008] for the benchmark functions and a constrained version of Trust Regions on Riemannian manifolds [Jaquier and Rozo, 2021] for the grasping experiments.

As in Kandasamy et al. [2016], we make the assumption that each of the M fidelities has an error that is bounded and is decreasing with increasing fidelity. The error of the m-th fidelity $||f^{(M)} - f^{(m)}||_\infty$ is assumed to be bounded by $\zeta^{(m)}$:

$$||f^{(M)} - f^{(m)}||_\infty \leq \zeta^{(m)} \qquad \forall m = 1, ..., M$$

with

$$\zeta^{(1)} > ... > \zeta^{(M)} = 0$$

Furthermore, in order to only have one variable ζ the variables $\zeta^{(1)}, \zeta^{(2)}, ..., \zeta^{(M-1)}$ are set as $(\zeta^{(1)}, \zeta^{(2)} ..., \zeta^{(M-1)}) = ((M-1)\zeta, (M-2)\zeta, ..., \zeta)$. This error bound of the fidelities is then used for the acquisition function φ_n that considers every fidelity at iteration n in the following way:

$$\varphi_n^{(m)}(\boldsymbol{x}) = \mu_{n-1}^{(m)}(\boldsymbol{x}) + \beta_n^{1/2} \cdot \sigma_{n-1}^{(m)}(\boldsymbol{x}) + \zeta^{(m)}, \quad \forall m \qquad \varphi_n = \min_{m=1,...,M} \varphi_n^{(m)}(\boldsymbol{x}) \tag{4.1}$$

The acquisition function is the sum of three terms. The first term $\mu_{n-1}^{(m)}(\boldsymbol{x})$ represents the exploitation part of the acquisition function as it considers the mean of the surrogate model at the current iteration. The next term $\beta_n^{1/2} \cdot \sigma_{n-1}^{(m)}(\boldsymbol{x})$ represents the exploitation part of the acquisition functions. The factor $\beta_n^{1/2}$ is chosen to weigh the emphasis of the uncertainty $\sigma_{n-1}^{(m)}(\boldsymbol{x})$ of the underlying surrogate model at the current iteration. The last term $\zeta^{(m)}$ adds the constant error term for lower fidelities. The acquisition function tells us which point to query next. Unlike other MF approaches, the fidelity that will be queried next is determined after

deciding which point is queried next. To determine which fidelity to query we check if the lower fidelity (simulation) at the determined new point to query $\boldsymbol{x_n}$ is certain enough. Specifically, the next fidelity queried is determined as:

$$m_n = \min_m \{m | \sqrt{\beta_n} \cdot \sigma_{n-1}^{(m)}(\boldsymbol{x_n}) \geq \gamma_n^{(\mathrm{m})}\} \tag{4.2}$$

$\gamma_n^{(m)}$ represents the threshold of certainty at iteration n of fidelity m. Both parameters have the subscript n to indicate that they are dependant on the current iteration because they are both updated at each iteration of BO. ζ and γ are initialized as 1% of the range of initial queries. The intuition behind $\zeta^{(m)}$ is an upper bound of the error of the fidelity m. We want to update it if the error is actually larger. To do this, at any time when we query a fidelity m we also check if $|f^m(\boldsymbol{x_n}) - \mu_{n-1}^{m-1}| \geq \zeta$. If this is the case we then check if $|f^m(\boldsymbol{x_n}) - f^{m-1}(\boldsymbol{x_n})| \geq \zeta$. If this is the case ζ is updated to twice the violation.

$\gamma^{(m)}$ is the threshold of fidelity m and decides if the surrogate model is certain enough for this fidelity. Therefore, if the MF setup is stuck in a fidelity m the parameter $\gamma^{(m)}$ is poorly set and we want to increase the threshold. Specifically, if the MF setup queries the fidelity m for $\frac{\lambda^{(m+1)}}{\lambda^{(m)}}$ times in a row the threshold γ is doubled. β is updated at every iteration and depends on the dimension of the problem d and the iteration n:

$$\beta_n = 0.2d \log(2n)$$

Algorithm 1 describes the pseudo code of our MF-GaBO algorithm on exactly two fidelities. An example of the MF-GaBO setup on $\mathbb{R}$ is displayed in Figure 4.1. The MF setup is being initialized with 3 points in the simulation and the experiment each. We can see that that until iteration 11 the MF setup almost exclusively queried the simulation. At the iteration 10 the threshold ζ was surpassed for the first time. Therefore, the experiment was queried.

Algorithm 1 MF-GaBO

budget $\leftarrow \Lambda$
$GP^{(\text{sim})} \leftarrow (\{\boldsymbol{x}_{\text{init}}\}, (\mathbf{0}, \sqrt{\boldsymbol{K}}))$ ▷ Initializing sim GP
$GP^{(\text{exp})} \leftarrow (\{\boldsymbol{x}_{\text{init}}\}, (\mathbf{0}, \sqrt{\boldsymbol{K}}))$ ▷ Initializing exp GP
$\zeta \leftarrow \gamma^{(\text{sim})} \leftarrow 1\% \cdot$ initial range
while budget ≥ 0 **do**
  Fit hyperparameters of all GP models using MLE on the MLL.
  $\boldsymbol{x}_n \leftarrow \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \varphi_n(\boldsymbol{x})$ ▷ Using Trust Regions on Riemannian manifolds
  if $\sqrt{\beta_n \cdot \sigma^2_{n-1}(\boldsymbol{x}_{\text{new}})} \leq \gamma_n^{(\text{sim})}$ **then** ▷ Query experiment
    $y_n \leftarrow f^{\text{exp}}(\boldsymbol{x_n})$
    if $|f^{\text{exp}}(\boldsymbol{x_n}) - \mu^{\text{sim}}_{n-1}| \geq \zeta$ **then**
      $y_{\text{n sim}} \leftarrow f^{\text{sim}}(\boldsymbol{x_n})$
      Add $y_{\text{n sim}}$ to GP^{sim}
      if $|y_n - y_{\text{n sim}}| \geq \zeta$ **then**
        $\zeta \leftarrow 2\zeta$
      end if
      budget $\leftarrow$ budget - λ^{exp}
      Add y_n to GP^{exp}
    end if
  else if $\sqrt{\beta_n \cdot \sigma^2_{n-1}(\boldsymbol{x}_{\text{new}})} > \gamma_n^{(\text{sim})}$ **then** ▷ Query simulation
    $y_n = f^{\text{sim}}(\boldsymbol{x_n})$
    if Simulation has been queried for $\frac{\lambda^{\text{exp}}}{\lambda^{\text{sim}}}$ consecutive times **then**
      $\gamma \leftarrow 2\gamma$
    end if
    budget $\leftarrow$ budget - λ^{sim}
    Add y_n to GP^{sim}
  end if
end while

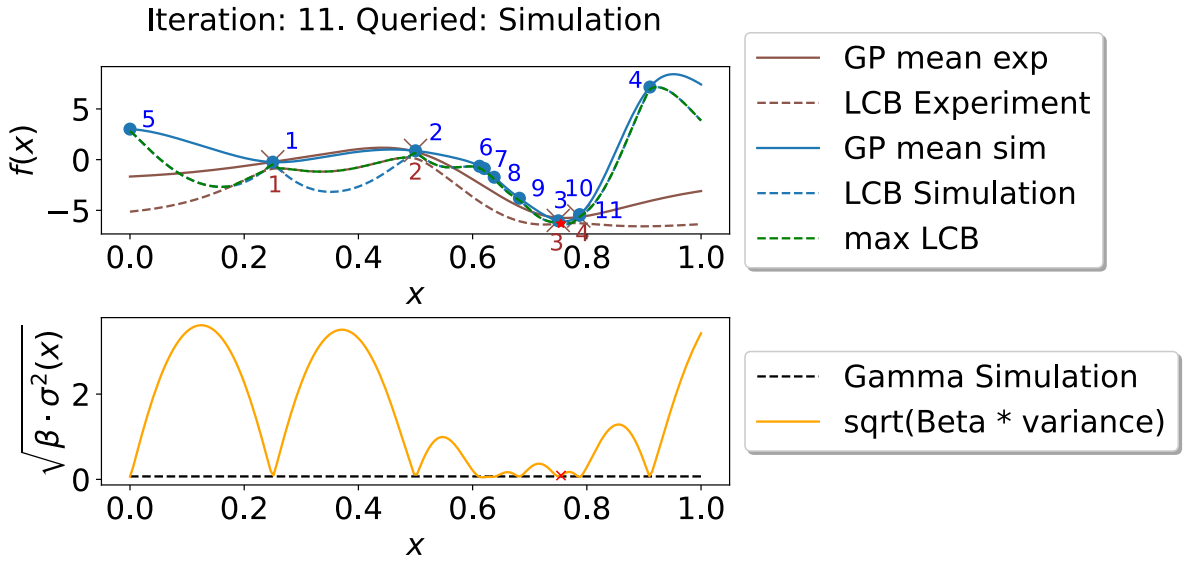


Figure 4.1: The top graph shows the two surrogate GP models of the Kandasamy MF setup applied to optimize the Forrester function. The bottom graph shows the decision of which fidelity to query next. In the top graph, the blue continuous line shows the mean of the simulation GP and the annotated numbers show the order in which the points got queried. The blue dashed line show the lower confidence bound of the simulation GP. The brown color represents the experiment GP analogously. The green dashed lined shows the acquisition function. The minimum of the acquisition function is shown with a red dot and corresponds to the next point to query. In the bottom graph, the yellow function shows $\sqrt{\beta \cdot \sigma^2}$ which is checked against ζ to determine which fidelity to query.

Chapter 5

Implementation

For this thesis we use the bullet[1] physics simulation. For all robot experiments we use the humanoid robot ARMAR-6 [Asfour et al., 2018] built by H2T at KIT. The robot is displayed in Figure 5.1. ARMAR-6 has two 8-DOF arms and an underactuated hand that uses tendons to pull the fingers for grasping. As software framework we use the ArmarX framework [Welke et al., 2013]. It allows us to communicate with the robot ARMAR-6 and the simulation that the ArmarX framework provides in the same way. Furthermore, it allows us to access various sensor data provided by either ARMAR-6 or the simulation in the same way. This was especially useful for the design of the objective function, described in detail in subsection 6.2.2. The ArmarX framework uses different layers of abstraction from abstract high level components down to hardware near low level components. The ArmarX framework structure is displayed in Figure 5.2. In this thesis we only use the high level *Robot Coordination* layer. There we incorporate our experiment as different components: `SetupExperiment` for putting the robot in neutral starting position, `GraspDishwasherHandle` for grasping the the dishwasher handle, `SkillEvaluation` for retrieving the data out of the robot memory and finally `ReleaseHandle` for automatically releasing the handle. The implemented interfaces are discussed in section 5.2. Because of the complex structure of the framework it is error prone. Furthermore, our implemented components can also crash unexpectedly and in order to prevent potential damage on the robot, e.g., fingers of the hand breaking we also have to supervise the execution and

[1] https://github.com/bulletphysics/bullet3

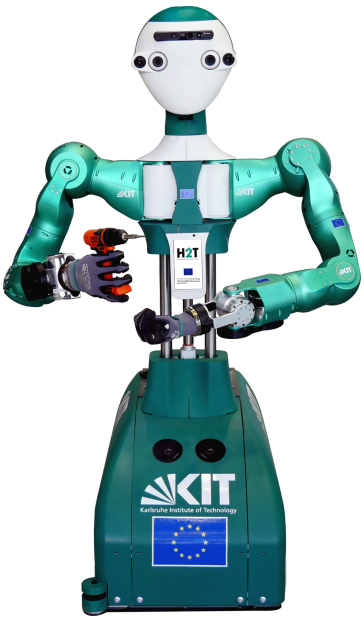


Figure 5.1: The robot ARMAR-6 [Asfour et al., 2019]

potentially manually interrupt the task. Therefore, our goal is to design robust and recoverable software. To follow these design principles we implement reload-ability of BO runs and robust error handling. Because of the intrinsic isolated structure of the ArmarX framework we also develop a file system communication discussed in the following section.

5.1 Implementation details

Robustness Because each component in each layer could crash due to human implementation errors the code needs to be robust. First, we make sure that if any component crashes we catch the corresponding errors, allow the user to fix the problem and then allow the user to continue the BO execution. A common example of this is that a component crashes, then the user restarts the crashed component and continues the BO run. Another measure we take is to save the state of the BO run in each iteration. This includes the data on which the surrogate models are trained, all hyperparameters of the surrogate model and meta information such as the current iteration. This is especially useful if a problem occurs that is no immediately fixable or if the experiment needs to be interrupted for some other reason.

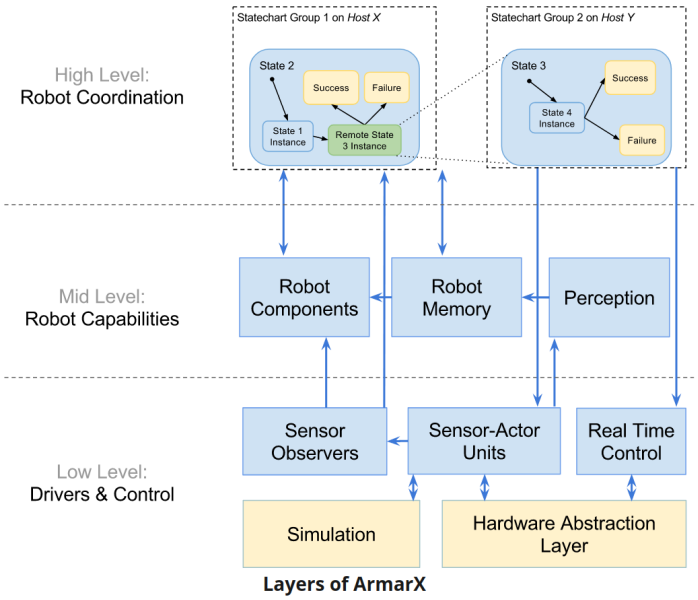


Figure 5.2: The structure of the ArmarX framework [Welke et al., 2013]

.

Multi-fidelity Convenience The ArmarX framework uses so called ArmarX profiles to manage port numbers. This is because ArmarX uses a specific port for the database that represents the robot memory and for the central Ice-Server that all components need to communicate with. Therefore one ArmarX profile only provides one python environment that is isolated from other environments. For MF-BO we need two environments though, one for each fidelity. Therefore, to make MF-BO have access to multiple environments we implemented a file system communication. Because at H2T all computers are connected and have a shared file system we can access the same file system with different environments. Here we implement a wrapper for the python environment which saves requests to the environment as files in a queue. The request is saved as json file containing the configuration which is then read by the wrapper on another computer and executed. Similarly, the wrapper also saves the computed result in the same queue. This way, MF-BO can access multiple environments across different computers.

5.2 Interface

All components that directly interact with ArmarX are written in C++ because of its speed. We use Ice[2] to provide a Python interface, in which we implement our BO setups.

[2]https://zeroc.com/products/ice

General structure In order to access the interface provided by the Ice components we implement a python environment. This python environment serves as interface for the BO implementations. When called it accesses the Ice interfaces to access the C++ interface. For each iteration, the python environment first calls accesses the `SetupExperiment` component to put the hand in a neutral starting position. Then the `GraspDishwasherHandle` component is used to grasp the handle. To do so, the environment provides two parameters:

1. The positional offset the right of the dishwasher in mm as float.
2. The quaternion that describes the orientation of the hand as list of floats.

The result of this remote procedure call is:

1. Whether the grasping of the dishwasher was successful as boolean.
2. A series of timestamps indicating the phases of the skill, e.g., when the hand reached the dishwasher door and starts moving upwards towards the door handle.

Directly after grasping the handle, the grasp is evaluated using the `SkillEvaluation` component. The `SkillEvaluation` component has several methods that allow the evaluation of:

1. Manipulability trajectory: Given a time interval the manipulabilty over the specified time interval is returned.
2. TCP pose trajectory: Given a time interval the TCP pose over the specified time interval is returned.
3. Forces and torques: Given a time interval the forces and torques over the specified time interval are returned.
4. Grasp distance: Given a time interval the distance between the hand and the dishwasher handle over the specified time interval is returned.

This is done directly after grasping the dishwasher handle because the working memory of the robot has a fixed size. Therefore, depending on the memory size and the frequency at which sensor data is written into the memory parts of the grasping duration could be overwritten. In our experiments we worked with a frequency of 25Hz which led to a time span of 1 minute of saved sensor data. With this information the objective function value is evaluated and returned to the caller of the environment. Finally, the `ReleaseHandle` component is used to safely

remove the hand from the dishwasher handle in a compliant manner. This is especially useful as it allows us to autonomously run the optimization except for potential failures. At each component if the execution is unsuccessful further steps are stopped and the execution is retried after the user signals the component that the problem has been resolved. Figure 5.3 shows the procedure as an UML activity diagram.

Having the abstraction of environments helps us run environments in parallel and allows wrappers of the environment which we use for the multi-fidelity setup. Figure 5.4 shows an UML diagram of the structure of our implementation.

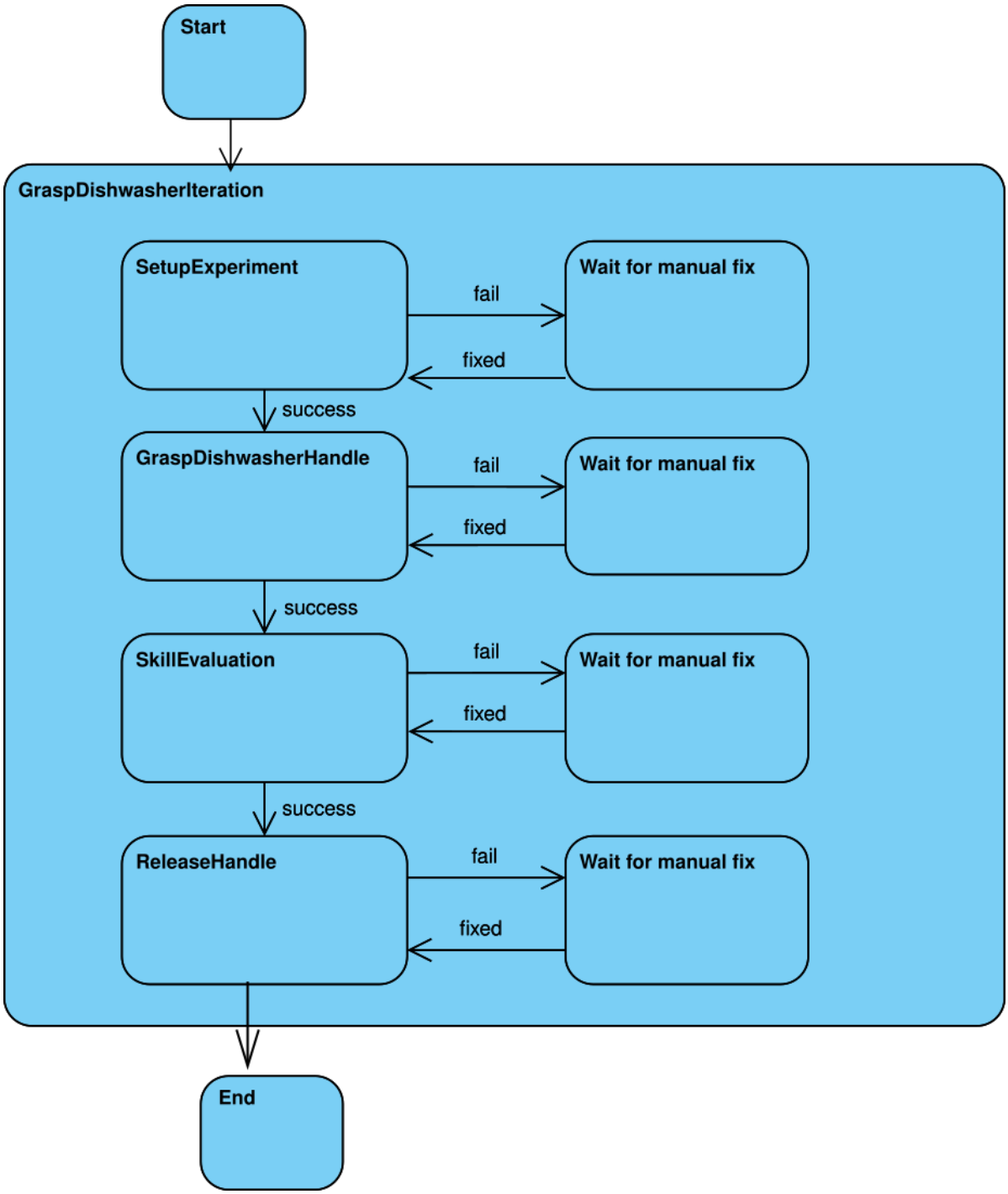


Figure 5.3: Activity diagram of an dishwasher grasping iteration.

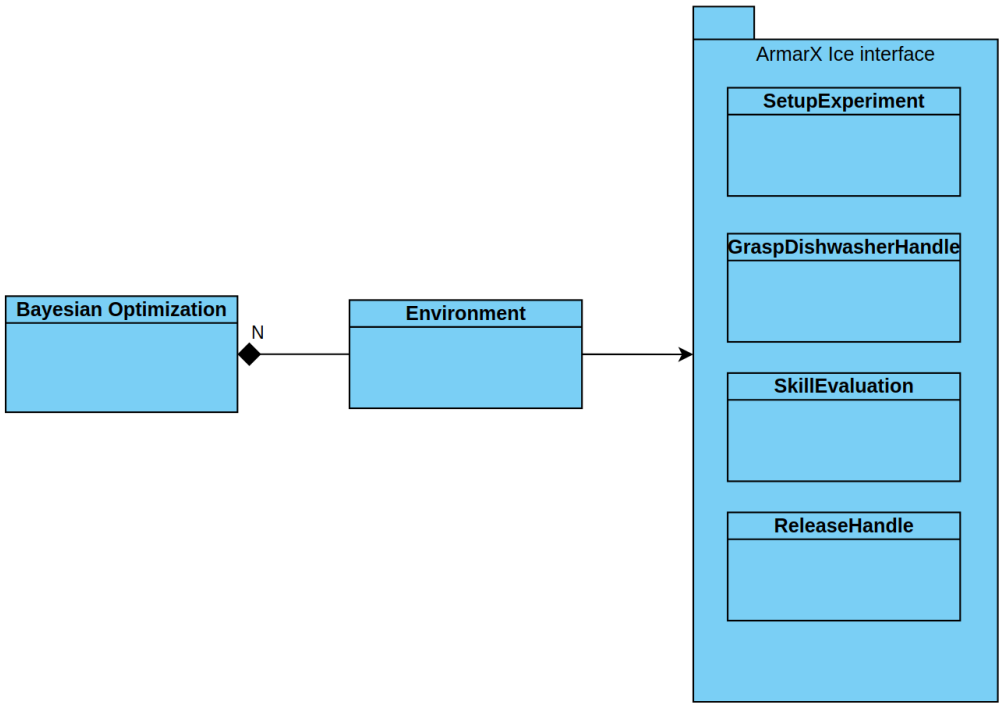


Figure 5.4: The structure of our implementation. The BO implementation can have several `Environments` which use the ArmarX interface. All the components `GraspDishwasherHandle`, `SkillEvaluation`, `SkillExecution` and `SetupExperiment` are part of the the ArmarX interface.

Chapter 6

Evaluation

6.1 MF GaBO benchmarks

In this experiment we want to test our MF GaBO algorithm. To do so, we compare the MF setup to a SF setup where we only query the higher fidelity. In this comparison we want to see if the extra fidelity helps the MF setup to converge faster. Additionally we want to test how the geometry-awareness affects our MF GaBO algorithm. Therefore, we compare MF GaBO to the Euclidean version of our MF GaBO algorithm which is the MF BO approach introduced by Kandasamy et al. [2016] which we will refer to as MF EuBO. Because Euclidean BO does not intrinsically constrain the values to have normalized length like GaBO we constrain the optimization to normalized values.

For this experiment we test our algorithm on two different commonly used multi-fidelity benchmark functions, the 2-dimensional Currin function and the 8-dimensional Borehole function [Xiong et al., 2013]. Both benchmark functions use two fidelities. In our experiments we defined the cost for a query on the lower fidelity (simulation) $\lambda_{\text{sim}} = 1$ and the cost for the higher fidelity (experiment) $\lambda_{\text{experiment}} = 10$. For better comparability in this benchmark we use 30 different seeds that determine the initialization points and average them. For both the Currin and Borehole benchmark function we compare three different setups for 150 iterations each:

- SF GaBO on the sphere manifold.
- MF GaBO on the sphere manifold for both fidelities.
- MF EuBO in Euclidean space for both fidelities.

6.1.1 Implementation details

The Currin function and its lower fidelity modification are defined as:

$$f_{\text{currin}}(x_1, x_2) = \left[1 - \exp\left(-\frac{1}{2x_2}\right)\right] \cdot \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}$$

$$f_{\text{currin lofi}}(x_1, x_2) = \frac{1}{4}[f_{\text{currin}}(x_1 + 0.05, x_2 + 0.05) + f_{\text{currin}}(x_1 + 0.05, \max(0, x_2 - 0.05))] +$$

$$\frac{1}{4}[f_{\text{currin}}(x_1 - 0.05, x_2 + 0.05) + f_{\text{currin}}(x_1 - 0.05, \max(0, x_2 - 0.05))]$$

for the domain $x_i \in [0, 1] \quad \forall i = 1, 2$

The Borehole function and its lower fidelity modification originally are defined as:

$$f_{\text{borehole}}(x) = \frac{2\pi T_u(H_u - H_l)}{\ln(r/r_w)\left(1 + \frac{2LT_u}{\ln(r/r_w)r_w^2 K_w} + \frac{T_u}{T_l}\right)}$$

$$f_{\text{borehole lofi}}(x) = \frac{5T_u(H_u - H_l)}{\ln(r/r_w)\left(1.5 + \frac{2LT_u}{\ln(r/r_w)r_w^2 K_w} + \frac{T_u}{T_l}\right)}$$

with $x = (r_w, r, T_u, H_u, T_l, H_l, L, K_w)^\intercal$ and the bounds:

$$r_w \in [0.05, 0.15]$$
$$r \in [100, 50000]$$
$$T_u \in [63070, 115600]$$
$$H_u \in [990, 1110]$$
$$T_l \in [63.1, 116]$$
$$H_l \in [700, 820]$$
$$L \in [1120, 1680]$$
$$K_w \in [9855, 12045]$$

Transformation of input domains In order to test MF GaBO, we project the benchmark functions onto the surface of a sphere. To do so, we define the base of the sphere manifold as $(1, \underbrace{0, \ldots, 0}_{d-1})^T$ and use its tangent space as the input for the benchmark functions. This yields a $d-1$-dimensional Euclidean space in which the benchmark function can be computed. The function is then projected onto the manifold with the exponential map. To ensure that the optimum is reachable for points $\boldsymbol{x}$ on the manifold mapped to the tangent space we transformed the function, so that the optimum is at the origin $(0, \ldots, 0)^{d-1}$ of the tangent space. For both benchmark functions we first take the absolute value of the logarithmic mapping $|\mathrm{Log}_{\mathrm{base}}(\boldsymbol{x})|$. From there, we applied a linear transformation so that the optimum would reside at the origin. The resulting benchmark functions are the following:

$$f_{\text{pre currin}}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} |x_1|/\pi \\ |x_2|/\pi \end{pmatrix} + \begin{pmatrix} 0.2165 \\ 0.0305 \end{pmatrix}$$

$$f_{\text{currin transformed}}(x_1, x_2) = f_{\text{currin}}(f_{\text{pre currin}}(x_1, x_2))$$

$$f_{\text{pre borehole}}\begin{pmatrix} x_1 \\ \vdots \\ x_8 \end{pmatrix} = \begin{pmatrix} |x_1|/\pi \cdot (0.15 - 0.05) \\ \vdots \\ |x_2|/\pi \cdot (12045 - 9855) \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

$$f_{\text{borehole transformed}}(x_1, x_2) = f_{\text{borehole}}(f_{\text{pre borehole}}(x_1, x_2))$$

The last factor of $f_{\text{pre borehole}}$ is chosen because the optimum without the factor lies at $[1, 0, 1, 1, 1, 0, 0, 1]^T$ of the hypercube $[0, 1]^8$. Therefore, to center to optimum in the origin we inverted some dimensions.

6.1.2 Results

In this benchmark we specifically want to compare SF GaBO to MF GaBO to see if the additional fidelity in our MF GaBO algorithm helps to converge faster. Furthermore, we compare MF GaBO to MF EuBO to see how geometry-awareness impacts the MF BO. Our finding is that the Multi-fidelity setups could find the the optimum faster than the Single-fidelity setups. Figure 6.1 shows the benchmark results for the Currin function. The figure clearly shows that MF GaBO outperforms SF GaBO and MF EuBO. Our interpretation is that the Multi-Fidelity setup is able to use the information from the lower fidelity to find the optimum in a faster manner. We also compared the the amount of queries in the lower fidelity (simulation) and in the higher fidelity (experiment) of the Multi-Fidelity setups in both Figure 6.1 and Figure 6.2 on the right plot, albeit not finding any significant interpretation.

We can see similar results in the benchmark results of the Borehole function, shown in Figure 6.2. The 8-dimensional benchmark function favours the Multi-Fidelity setup more than the 2-dimensional Currin function. In this benchmark we can clearly see that the Multi-Fidelity setups converge at a much higher rate in the first 100Λ spent. Our interpretation is that the Multi-Fidelity setup is able to query the lower fidelity (simulation) at a cheap cost and explore sufficiently in the lower fidelity.

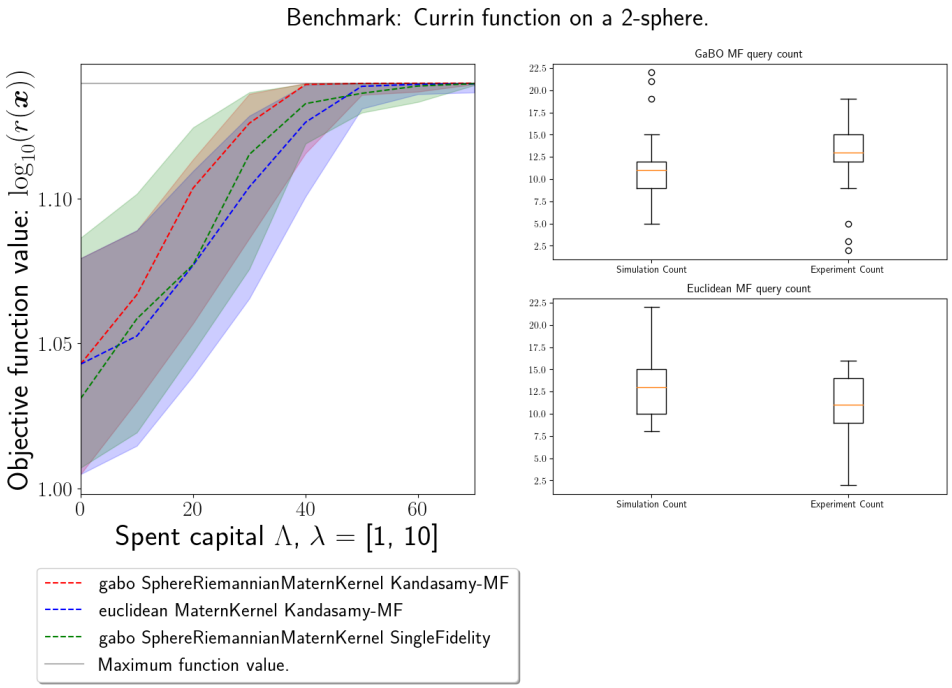


Figure 6.1: The left plot shows the benchmark results of the Currin function of the 3 different approaches: SF GaBO (green), MF GaBO (red) and MF EuBO (blue). The dashed line is the median over all seeds and the shaded area around is the range between the first and third quartile. The right plot shows the amount of queries the Multi-Fidelity setups queried in the higher fidelity (experiment) and in the lower fidelity (simulation). We can see that MF GaBO converges faster and with less variance than MF EuBO and SF GaBO.

After this initial phase we can see both Single-Fidelity and Multi-Fidelity Geometry-aware BO converging better than the Euclidean BO. Our interpretation is that Geometry-aware BO is able to to use the underlying geometry to better exploit available observations for the exploitation/exploration paradigm of the acquisition function and therefore converging faster and better than EuBO.

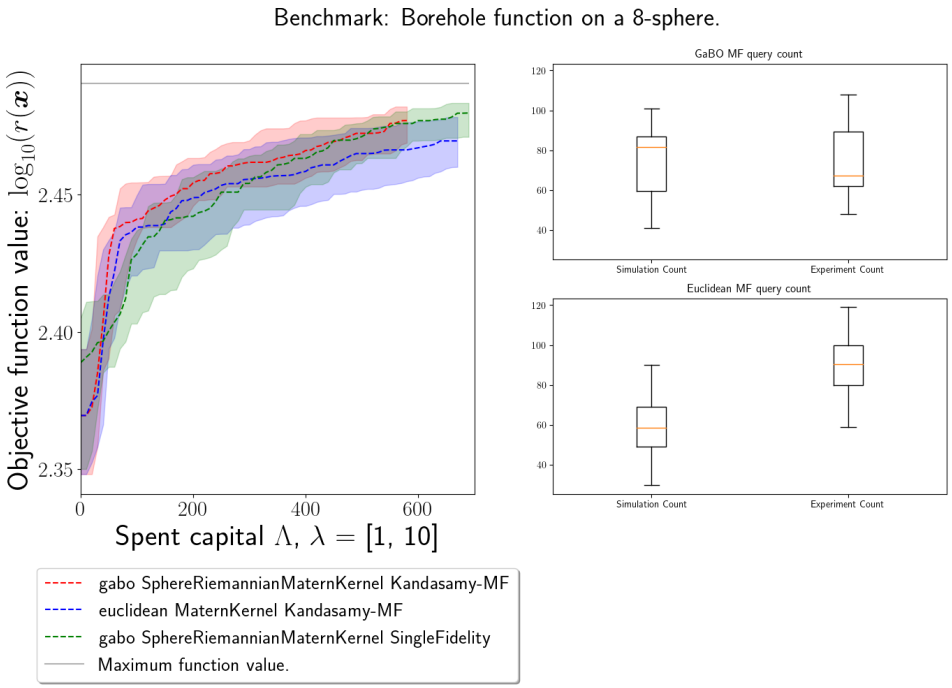


Figure 6.2: The left plot shows the benchmark results of the Borehole function of the 3 different approaches: SF GaBO (green), MF GaBO (red) and MF EuBO (blue). The dashed line is the median over all seeds and the shaded area represents the area between the first and third quartile. Both MF GaBO and MF EuBO end early, because the spent a lower capital by querying some iterations in the simulation.
The right plot shows the amount of queries the Multi-Fidelity setups queried in the higher fidelity (experiment) and in the lower fidelity (simulation). We can see that MF GaBO converges faster and with less variance than MF EuBO and SF GaBO.

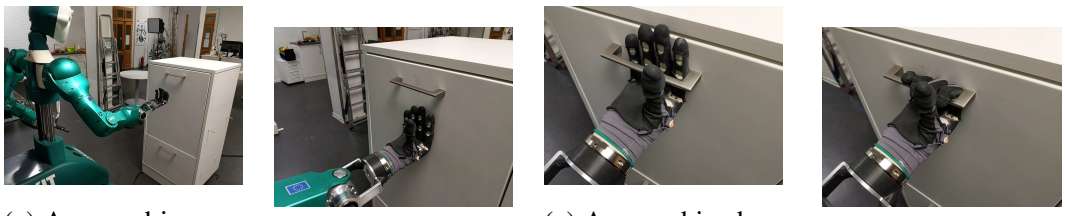
6.2 Real-world grasping experiments

6.2.1 Simulation and ARMAR-6 Implementation

Our experiment consists of two main objects: the robot and the dishwasher. The robot is placed in front of the dishwasher at a certain distance to grasp the dishwasher. We assume that the position of the dishwasher and the position of robot are known beforehand.

We divided the grasping of the dishwasher handle into several skills.

1. Put the hand in a predefined shape.



(a) Approaching door

(b) Door contact

(c) Approaching handle

(d) Grasping

Figure 6.3: Phases of Experiment shown by ARMAR-6.

2. Move the hand into a predefined pose in front of the dishwasher. In this phase the orientation and x-offset chosen from BO is considered.

3. Establishing door contact with the hand. In this phase the hand moves in a straight line towards the dishwasher door. Upon a fixed threshold of forces and torques is measured at the wrist the robot moves on to the next phase.

4. Approaching door handle. In this phase the hand moves straight upwards towards the door handle. Similarly to the previous phase, this phase also stops at a fixed threshold measured at the wrist sensor.

5. Grasping the dishwasher handle. This is the last phase. Here the robot hand simply closes the hand.

After this chain of phases the robot automatically resets to its original pose implemented by two different skills. Some of the phases are shown in Figure 6.3.

Optimization To optimize this skill we designed an objective function that considers several criteria such as manipulability, distance measures, forces and torques. The details of the design process are discussed in subsection 6.2.2.

Similarly to section 6.1, we compare MF GaBO, SF GaBO and MF EuBO. There are some differences in the optimization though. In the grasping experiment we optimize over rotation around the x-axis, y-axis and one positional axis which are shown in Figure 6.4. We don't optimize over the z-axis because we believe the neutral position is already optimal. Therefore, we optimize over the product of manifolds of 1-dimensional Euclidean space and the 4-dimensional 3-sphere, whereas in section 6.1 we only optimize on the sphere manifold. To resemble the manifold product space a product kernel is used consisting of a sphere Matérn kernel and a Euclidean Matérn kernel. To avoid overfitting, the lengthscale parameter Θ of the Matérn kernel

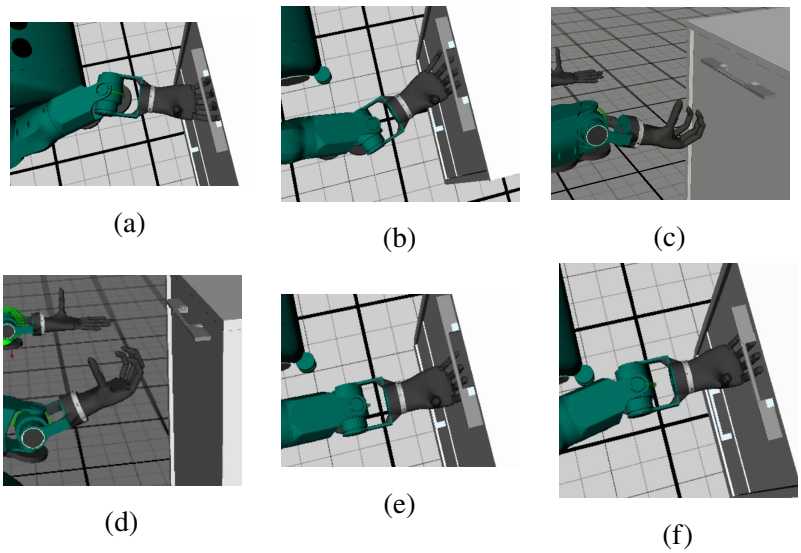


Figure 6.4: Rotation and position axes optimized in our dishwasher grasping experiment. (a) and (b) show the rotation over the x-axis. (c) and (d) show the rotation over the y-axis. (e) and (f) show the position over the y axis.

introduced in section 2.2 is shared across both kernels. Secondly, in our grasping experiments we constrain the range of available orientations to a roll and pitch range of 30° over both axes, 15° in the positive direction and 15° in the negative direction. In Figure 6.4 the axes optimized over are shown. For both Single-fidelity and Multi-fidelity setup we use both GaBO and EuBO to compare results which are discussed in subsection 6.2.3, resulting in six different setups:

1. SF Simulation GaBO
2. SF Simulation EuBO
3. SF ARMAR-6 GaBO
4. SF ARMAR-6 EuBO
5. MF GaBO
6. MF EuBO

Simulation Additionally, we also test the simulation separately to get insight on how much information the simulation offers. In the simulation we use a 3D mesh model of a dishwasher to represent the dishwasher. Because the collision model of the 3D mesh model of the dishwasher

did not behave as expected, e.g., in the simulation the hand would not fit into the dishwasher and sometimes would collide with the top of the dishwasher, we adjusted the dishwasher model to represent the real dishwasher more closely. Specifically, we changed the collision box of the the simulation model by removing the top of the dishwasher and replacing the handle with primitives that closely resemble the original handle. The top of the dishwasher is removed because in the simulation model the robot hand collides with the dishwasher roof before the grasping of dishwasher handle is triggered. The resulting collision model is portrayed in Figure 6.7 (c).

Positioning The grasping implementation assumes the position of the robot and the dishwasher is known beforehand. Therefore, we set the position of the robot and the dishwasher manually. We determined the ideal coordinates on ARMAR-6 simply by placing the robot such that the neutral execution, e.g., no positional offset and no rotation, of the grasping skill was centered. To reproduce this position in experiments on different days we marked the floor with tape. We then manually navigated the robot to its correct position and manually put the dishwasher on its correct position. To set the robot in the correct position at initialization in the simulation we use the same coordinates as on the robot ARMAR-6. This is possible because both simulation and ARMAR-6 use the same environment.

Hand Behaviour ARMAR-6 uses an underactuated hand. It uses tendons to pull the fingers to grasp. This results in the robot being able to produce torques to close the hand but it can only actively open the hand with minimal torques. Furthermore, when the hand is pressed against the dishwasher door and the lower joints close the upper joints closer the the fingertips automatically open. This behavior is shown in Figure 6.5. In contrast, in the simulation every joint of the hand is position controlled. Therefore, each joint can apply the same torques in both directions. Because our skill works by pressing the hand against the dishwasher door on ARMAR-6 so that the hand straightens we mimic this behaviour. To do so, we limit the maximal torques that the joints of the hand in the simulation can apply to be able to form the hand by pressing it against the dishwasher door. Additionally, we check the angle of the lowest joint. Therefore, if the hand has been bent and the lowest joint has passed a threshold the upper joints are opened.

Multi-fidelity setup In the Multi-fidelity setup we initialize the simulation surrogate model with 2 points and the experiment surrogate model with 1 initial point. For better comparability

(a) (b)

Figure 6.5: Underactuated hand behaviour of ARMAR-6. (a) shows the hand before being pressed against the dishwasher door. (b) shows the hand straighten up after being pressed against the dishwasher door and gently being opened.

we use run all BO setups with different seeds that determine the initialization points and average them. Since we used the same seeds for all BO setups: MF, SF simulation and 1 SF ARMAR-6, we used previously measured initial points from SF GaBO and SF EuBO ARMAR-6 experiments as initial points for both MF setups for consistency. SF GaBO and SF EuBO yielded different values for the same configuration because there is noise in the ARMAR-6 experiments. Therefore, we randomly chose between those two to fill the initial points of the experiment surrogate model for both MF setups.

6.2.2 Objective function design

The objective function is the cornerstone of any optimization. In our approach we consider several different criteria that we evaluate as weighted sum.

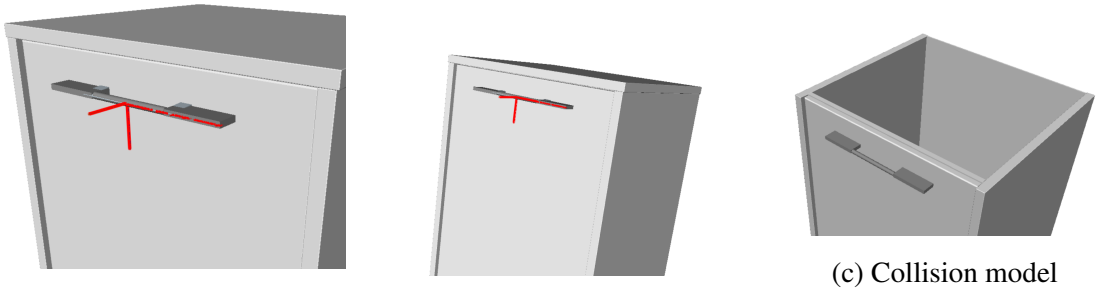
Manipulability The force manipulabilty ellipsoid [Yoshikawa, 1985] tells how much force the robot can apply in different directions. We take this after the grasping task at the hand of the robot because we want the robot to be able to open the dishwasher afterwards. We consider the force manipulability measure m by evaluating the force manipulability ellipsoid $\boldsymbol{M}$ for all positional axes of the hand, after the robot grasps the dishwasher handle. To get a scalar measurement out of $\boldsymbol{M}$ we take the sum over all eigenvalues: $m = \sum_{i=1}^{N} \lambda_i$ with λ being eigenvalues of $\boldsymbol{M}$ and N being the rank of $\boldsymbol{M}$. The manipulability term m is negated in the objective function because the objective function is minimized and we desire a high manipulability after grasping the dishwasher handle. In contrast, we desire low forces and distances.

(a) Side open (b) Side-top open (c) Side closed

Figure 6.6: TCP coordinate system used for distance measure.

Distance We use two different distance measurements: the distance between the hand and the handle after grasping d and what we call the impedance distance d_i. Both measurements are taken in millimeters (mm). We let the robot grasp the dishwasher handle in a compliant way. Therefore, when there are strong forces and torques on the hand before grasping and the hand enters the compliant mode, the hand moves a tiny amount to compensate. The resulting distance corresponds to the impedance distance. Thus, we want the impedance distance d_i to be small such that the hand does not move a lot after grasping. The coordinate systems used to measure the distance are shown in Figure 6.6 and Figure 6.7 (a) and (b).

Forces and Torques Forces and torques measured at the wrist of the robot hand in mN. The forces f_x, f_y, f_z and torques t_x, t_y, t_z are measured in the time interval between the start of approaching the door handle and the start of grasping the dishwasher handle. The coordinate system used for force and torque measurements is displayed in Figure 6.10. For both forces and torques all three axes x, y and z are considered. Depending on whether the experiment or the simulation is queried we weigh the forces and torques differently by a factor. For forces this factor fid_f is 1 if the experiment is being queried and $\frac{1}{6}$ otherwise. For torques this factor fid_t is 1 if the experiment is being queried and $\frac{1}{20}$ otherwise. This is because the forces and torques measured in the simulation differ from the real world experiment by a significant factor. Even though, the sensor should save the data with 25hz frequency, this was not consistently the case, as we worked on a robot which has side effects. To tackle this issue, we normalize both forces and torques by the amount of data entries that were actually written by the sensor to the robot memory l_t. In order to get a number that is better humanly readable we also add a factor $c = 0.16e^6$. This results in the following formulas for evaluating forces and torques.



(a) Coordinate System front (b) front side (c) Collision model

Figure 6.7: Dishwasher model used for experiments.

Thought process In order to understand the forces and torques during the dishwasher grasp we executed 6 different configurations and compared them:

1. Neutral position: no positional offset and no rotation of the hand.
2. 40mm offset to the left. The hand does not go inside the handle because the pinky finger collides with the handle.
3. 40mm offset to the right. The hand does not go inside the handle because the index finger collides with the handle.
4. 15 degree rotation downwards (rotation around y-axis in Figure 6.10). The finger tips get stuck on the dishwasher door and we manually abort the grasping execution to prevent damage of the hand.
5. 15 degree rotation to the right (rotation around the x-axis in Figure 6.10). The index finger is on the edge of the dishwasher handle and slips in.
6. 15 degree rotation to the right (rotation around the x-axis in Figure 6.10) and 15mm offset to the left (translation in the direction of the y-axis in Figure 6.10). The hand slips in without complications and barely touches the left end of the dishwasher handle.

Specifically we compare (5) and (6) to see how the finger barely slipping in affects the forces and torques. Additionally, we compare to (2) which we consider a failure as the hand misses the dishwasher handle. In Figure 6.8 we can see the comparison of the three configurations (2), (5) and (6). We can see that configuration (2) does not record negative y-axis forces when establishing door contact. This is because the hand is pointed straight forward. In contrast, both configuration (5) and (6) produce about 10mN when establish door contact because the hand is

rotated to the right. When grasping the handle configuration (5) produces about 10mN because the hand is being pushed in by the handle. Similarly, in configuration (6) the hand is pushed in by the handle from the left side and produces about 25mN. We can see almost the same results in Figure 6.9 because the torques around the x-axis and the forces in y-axis behave very similar. Because we do not want any forces caused by the hand touching the handle we emphasize forces in the y-axis direction and torques around the x-axis by giving the bigger weights.

$$f_j = \text{fid}_f \cdot c \cdot \frac{\sum_{t_{\text{start}}}^{t_{\text{end}}} |f_j|}{l_t} \qquad \forall j \in \{x, y, z\} \tag{6.1}$$

$$t_j = \text{fid}_t \cdot c \cdot \frac{\sum_{t_{\text{start}}}^{t_{\text{end}}} |t_j|}{l_t} \qquad \forall j \in \{x, y, z\} \tag{6.2}$$

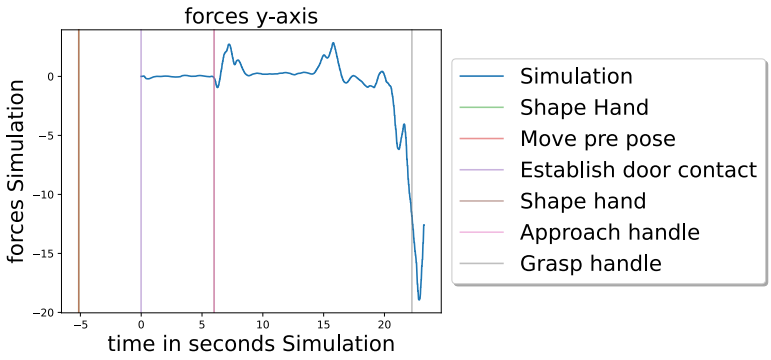
Objective function The resulting objective function that we wish to minimize is defined as:

$$f(d, d_i, f_x, f_y, f_z, t_x, t_y, t_z) = d + 0.2d_i - m + 0.025f_x + 0.05f_y + 0.0125f_z + 0.45t_x + 0.25t_y + 0.25t_z$$

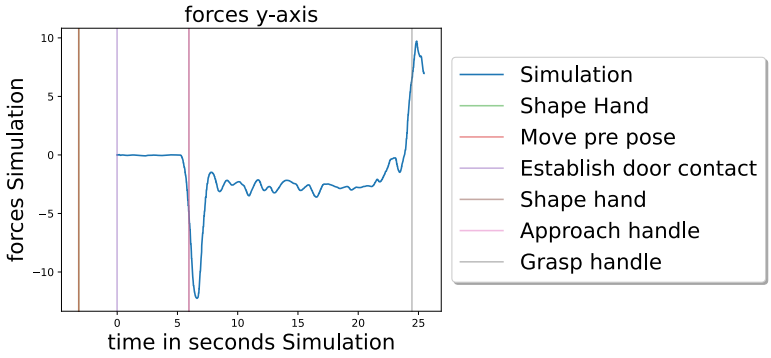
6.2.3 ARMAR-6 experiments

Single Fidelity In the Single-fidelity setup we compare GaBO and EuBO on the real setup only. Figure 6.11 shows both SF GaBO and SF EuBO along with the Multi-fidelty setups. We only compare the queries in the experiment fidelity. We can see GaBO (green) and EuBO (red) performing similarly well with GaBO converging slightly faster and with lower variance than EuBO. The different starting points of GaBO and EuBO are due to the experiments on ARMAR-6 having noise compared to the benchmark functions. The best configuration found is a slight offset to the right of 10.5mm with a rotation of 8 degrees around the x-axis (upwards towards the ceiling) and a rotation of -4 degrees around the y-axis (towards the right of the dishwasher). This configuration yielded the objective function value 29.8.

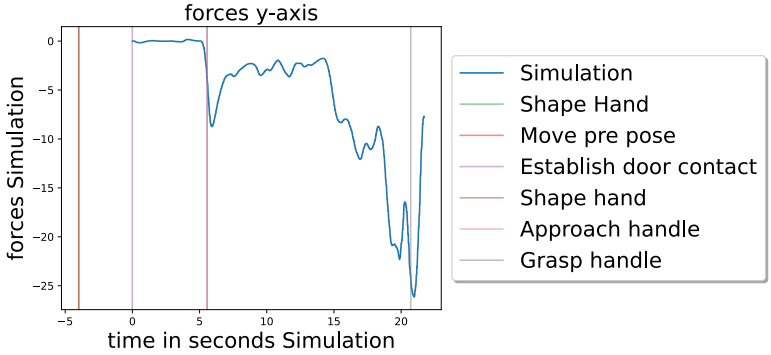
Multi-fidelity Figure 6.11 shows the comparison between the MF and the SF setups. In the ARMAR-6 experiments the MF setup showed no improvement over the SF setups. We can see that both Euclidean and Geometry-aware SF setups perform better than both MF setups. The best found configuration over all SF setups yielded an objective function value of 29.8. The best found configuration over all MF setups yielded the objective function value of 40.2. Similarly



(a) Configuration (2)

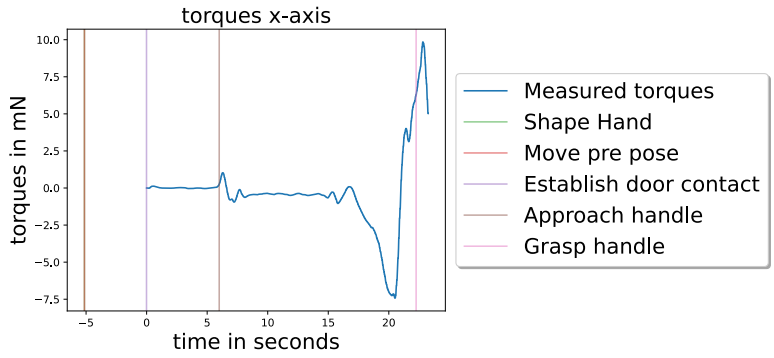


(b) Configuration (5)

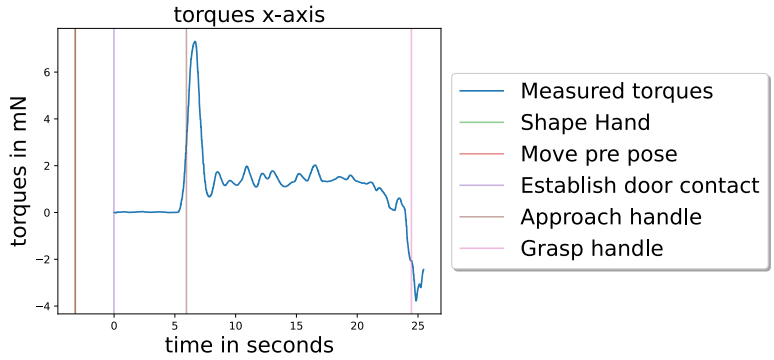


(c) Configuration (6)

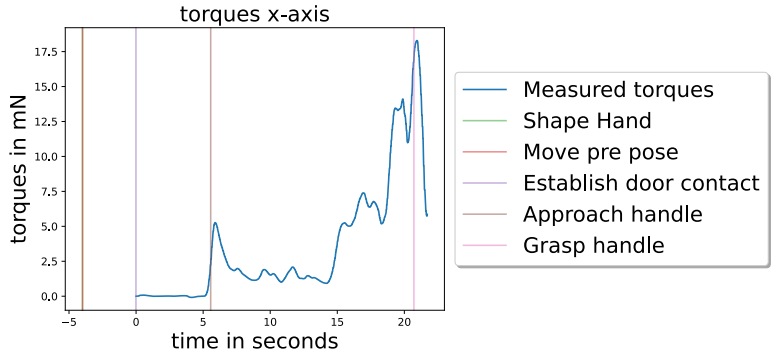
Figure 6.8: Forces in y-axis direction of configuration (2), (5) and (6). We can see that configuration (2) does not record negative y-axis forces when establishing door contact. This is because the hand is pointed straight forward. In contrast, both configuration (5) and (6) produce about 10mN when establish door contact because the hand is rotated to the right. When grasping the handle configuration (5) produces about 10mN because the hand is being pushed in by the handle. Similarly, in configuration (6) and (2) the hand is pushed in by the handle from the left side and produces negative forces in the y-axis direction.



(a) Configuration (2)



(b) Configuration (5)



(c) Configuration (6)

Figure 6.9: Torques around the x-axis of configuration (2), (5) and (6). Similarly to Figure 6.8, we can see that configuration (2) does not record much of x-axis torques when establishing door contact. This is because the hand is pointed straight forward. In contrast, both configuration (5) and (6) produce about 5mN when establish door contact because the hand is rotated to the right. When grasping the handle configuration (5) produces about -4mN because the hand is being pushed in by the right side of the handle. Similarly, in configuration (2) and (6) the hand is pushed in by the handle from the left side and produces positive torque.

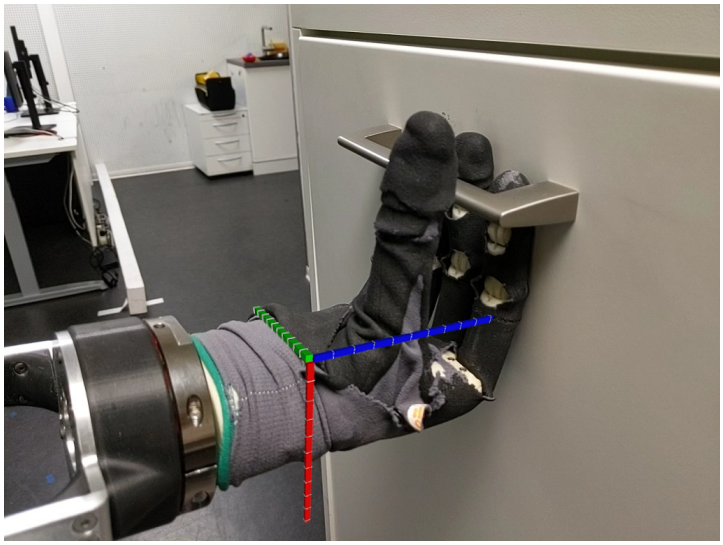


Figure 6.10: Coordinate system of the force measurements. The x-axis (red) points downwards. The y-axis (green) points sidewards to the left. The z-axis (blue) points in front of the hand towards the dishwasher.

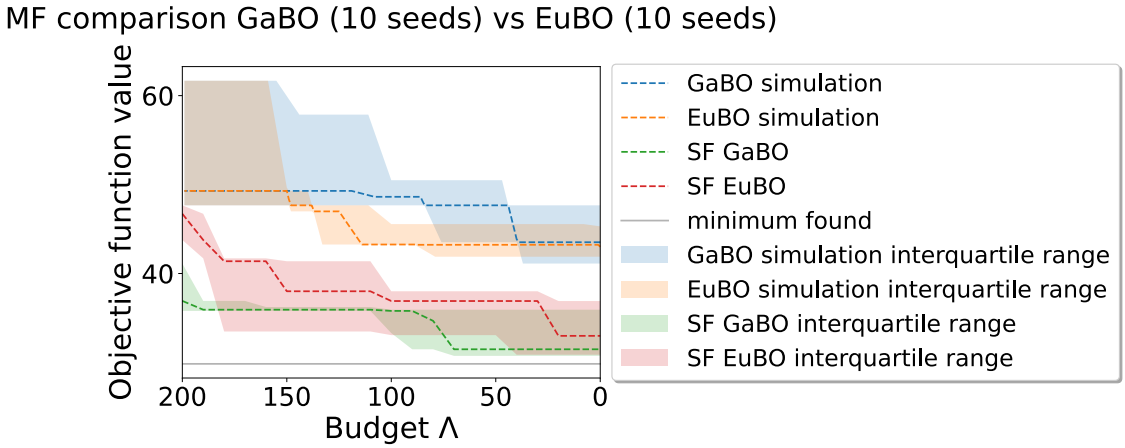


Figure 6.11: Comparison of MF GaBO (blue), MF EuBO (orange), SF GaBO (green) Armar-6 and SF EuBO Armar-6 (red). The dashed lines represent the according median and the shaded area is the corresponding interquartile range which is the range from the first quartile to the third quartile. The x-axis represents the budget Λ that is left to spend. The y-axis represents the best objective function value found so far in the highest fidelity (experiment).

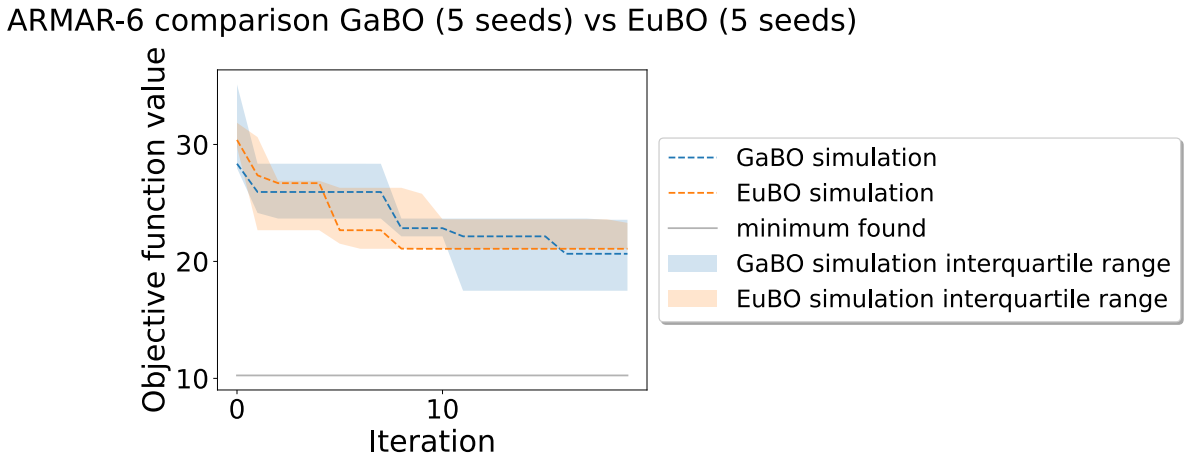


Figure 6.12: Comparison of EuBO (orange) and GaBO in simulation. The grey line indicates the global minimum found over both EuBO and GaBO.

to the best found SF configuration, the best found MF configuration also has a shift to the right of 18.0 mm, a rotation of 8.7 degrees around the x-axis (upwards to the ceiling) and a rotation of -6.9 degrees around the y-axis (towards the right of the dishwasher). We hypothesize that the simulation does not grant sufficient information to improve the MF BO. In section 6.2.3 we verify this hypothesis and test the simulation in a more detailed manner. We can also see that the MF setups start to query the higher fidelity (experiment) starting after around 40 iterations at a budget of $\Lambda = 160$ left, because then the first changes in the best found objective function value in the experiment start to show.

One might also question why all 4 setups do not start at the same best found objective function value. For SF GaBO and SF EuBO this is because of the noise of the experiment on ARMAR-6. Both MF setups start the same because we used results from SF runs as initial points as discussed in section 6.2.1.

Simulation experiments We previously hypothesized that the simulation does not grant sufficient information to improve the MF BO. In order to look further into the simulation we conduct three additional experiments:

1. A comparison between GaBO and EuBO in simulation.
2. An EuBO run with 200 iterations to test the convergence in the simulation.
3. A reproducability test of the minimum found in (1).

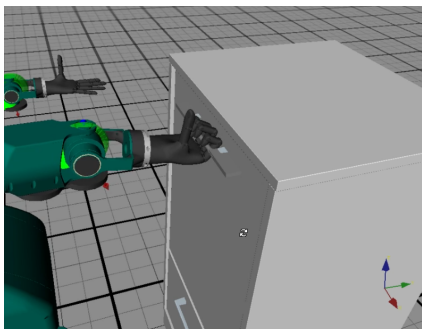


Figure 6.13: The execution of the configuration that yielded the minimum objective function value.

In the simulation we tested GaBO and EuBO each with 5 different seeds. In the comparison in Figure 6.12 we average over the 5 seeds and consider the median and the interquartile range. We can see that both GaBO and EuBO roughly perform the same. Interesting is the grey line at the bottom which represents the minimum found over both GaBO and EuBO. The minimum configuration for this result has the offset to the right of 22.5130, a rotation around the x-axis (downwards towards the floor) of $-7.67°$ degrees and a rotation around the y-axis (towards the right of the dishwasher) of $-2.39°$ degrees. While the second orientation around the y-axis is similar to the optimum found on ARMAR-6 the orientation around the x-axis is inverted. Our hypothesis is this is due to the friction of the fingertips of the hand of ARMAR-6. The fingertips have a rubber surface with high friction to grasp objects. In the simulation we can not replicate this because friction is not configurable. When the hand is tilting downwards in the simulation results in the finger being straighter which is desirable. In contrast, on ARMAR-6 the friction of the fingertips is so high that the hand gets stuck on the dishwasher.

We tested the configuration that resulted in the found minimum with an objective function value of 10.25 and found out that the result is not reproducible. Specifically, we ran the same configuration for 30 iterations. A plot of this reproducibility test is shown in Figure 6.14. In the 30 iterations the minimum objective value was 20.92 with a mean of 27.35 and a standard deviation of 3.38. For the same configuration this variance is significant. Furthermore, the initial value of 10.25 could not be reproduced. Therefore, we conclude that the simulation is not reliable enough. Figure 6.13 shows the simulation running the configuration that yielded the minimal objective function value.

To understand which term of the objective function contributes to this noise we look at term individually. Figure 6.15 (a) shows the individual plots of each term of the objective function weighted as in the objective function. Figure 6.15 (b) shows the relative variance of each term.

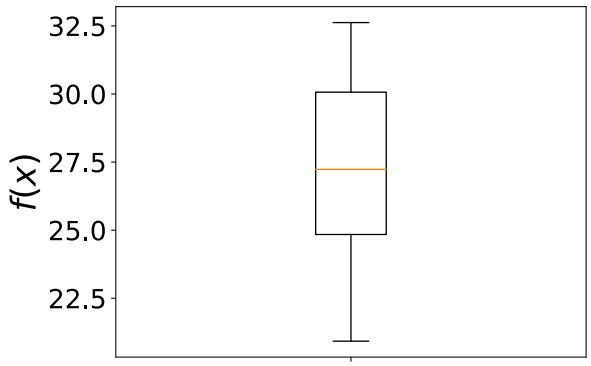


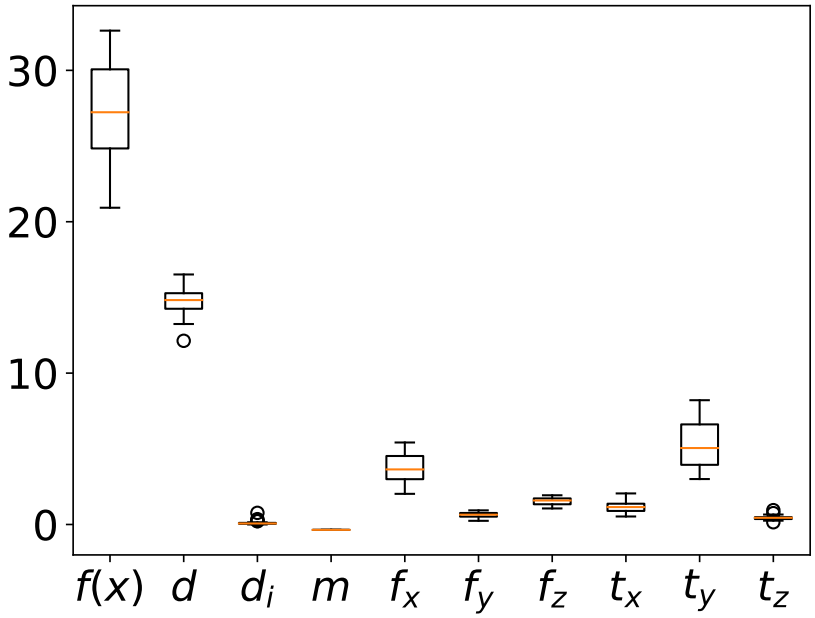
Figure 6.14: Reproducibility test of the found minimum in the simulation experiments. The boxplot shows the objective function values of 30 executions of the same configuration in the simulation.

We can see that even though f_x and t_y have the most absolute variance, all force and torque measurements are similarly noisy. Our interpretation is that our usage of the simulation is too noisy in this regard. A solution for future work is to use other simulations or configure the simulation to a certain degree that it is reliable enough.

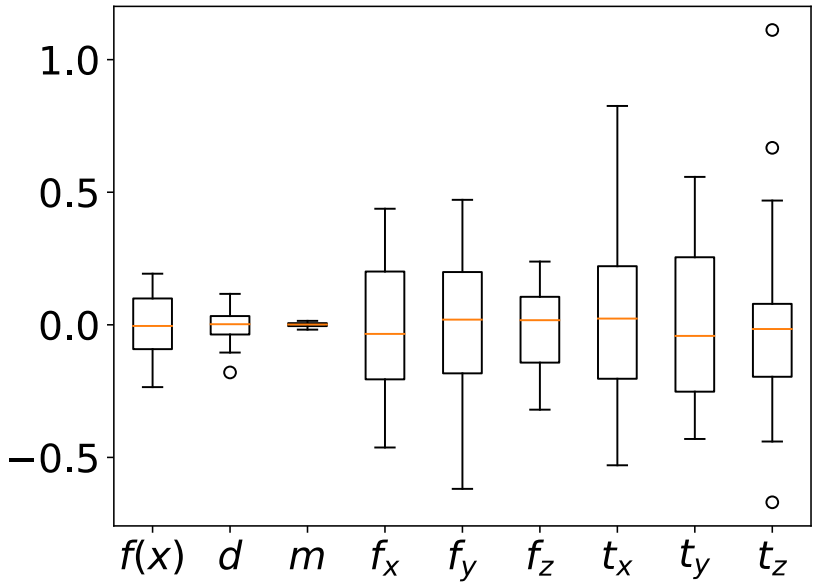
In order to further test how suitable the simulation is we ran a SF EuBO run of 200 iterations in the simulation. The results are shown in Figure 6.16. The first plot shows the so far best found objective function value over all iterations. Even though we can see steady improvements in the first plot our interpretation is that in the simulation BO does not converge. If BO would converge we would expect:

- BO to exploit around recently found good objective function queries.
- An overall steady improvement of the queried objective function results.

In Figure 6.16 (b) we do not see an overall steady improvement of the queried objective function results. We thus conclude that the simulation is not reliable enough.

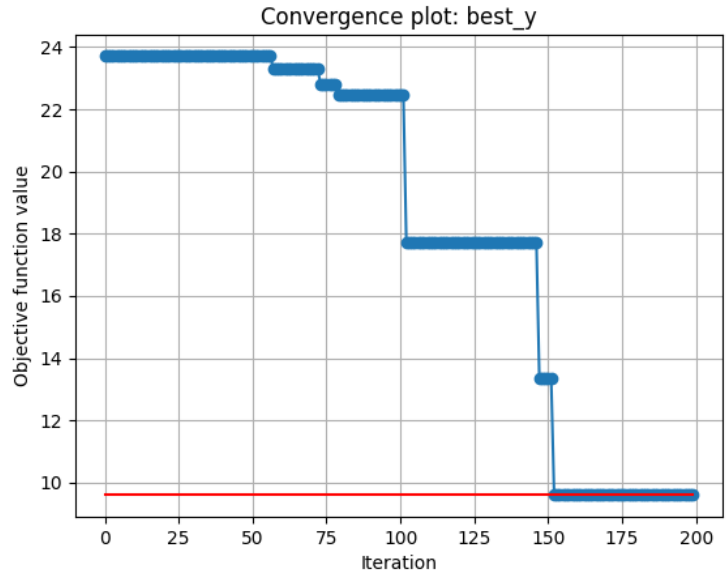


(a)

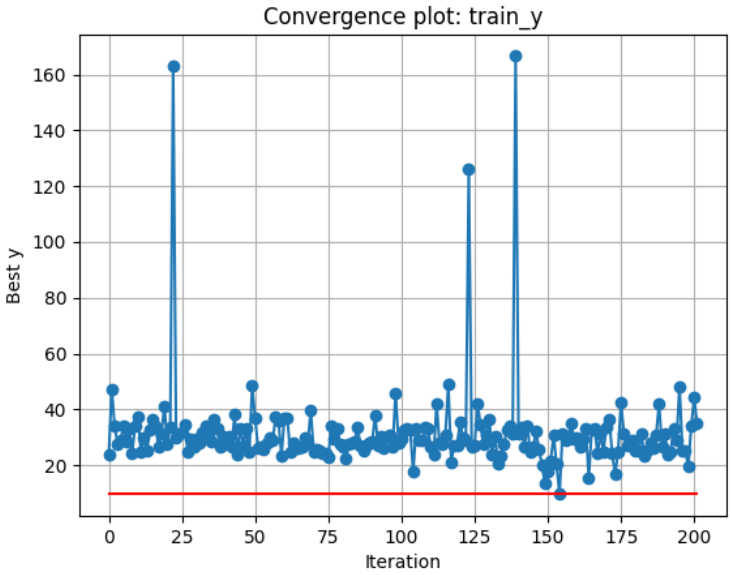


(b)

Figure 6.15: Figure (a) shows the objective function value and all weighted terms of the objective function value of the same configuration for 30 iterations. Figure (b) shows the data from (a) but normalized, e.g., first we subtracted the mean and then divided the signal by its mean. In this plot we can see that all force and torque measurements are noisy.



(a)



(b)

Figure 6.16: Plot of convergence test in simulation. (a) shows the objective function values of the so far best queried iteration of the BO run. (b) shows the objective function value of the latest query. On both plots the red line indicates the minimum found over all iterations.

Chapter 7

Conclusion and Future Work

Conclusion In this thesis, we presented a new MF GaBO algorithm which allows the incorporation of a lower fidelity as an additional information source and uses geometry-aware kernels and optimization methods.

We evaluated our algorithm in chapter 6 and show that MF GaBO converges faster and with less variance than MF EuBO and SF GaBO in the benchmark of the Currin and the Borehole function. Specifically, in the 8 dimensional Borehole function we see the extra information gained from the simulation fidelity, allowing MF BO to explore in the lower fidelity.

We then evaluated our algorithm in a grasping experiment to test MF GaBO. We tested the MF approach on two fidelities: the robot ARMAR-6 and a simulation where we simulate the grasping experiment with ARMAR-6. This results in no significant advantage over SF GaBO on ARMAR-6. We then analyzed the simulation in more detail to conclude the simulation was not providing sufficient information to improve MF GaBO over SF GaBO.

Future Work Bayesian Optimization and its derivatives MF BO and GaBO are emerging in robotics. We showed that in benchmarks Multi-fidelity setups can have an advantage over Single-fidelity setups. We believe that the information of lower fidelities can be utilized especially in robotics where various fast computable resources such as simulations exist.

In this thesis our use of the simulation was a limitation. However, in general lower fidelities can be incorporated in BO as shown in the benchmark results. We think that despite of our results, MF approaches in robotics needs more research. Especially because many different approaches to MF BO exist that can be used with many different lower fidelities.

Throughout the thesis we saw that GaBO performs better than EuBO, specifically in the high dimensional Borehole benchmark function. There, we can see GaBO converging faster and with more certainty. This difference was not as prominent in the grasping experiments. This could be because we only optimize over five dimensions: the 4 dimensions of the quaternion and 1 dimension of the positional offset instead of the 8 dimensional Borehole benchmark function. Additionaly, because we constrain the optimization of the hand rotations to a range of 30 degrees for both roll and pitch and keep a fixed yaw, the surface of the sphere that we actually optimize over is rather small and therefore the information gained from being geometry-aware is less.

We believe if the underlying geometry is known it can and should be incorporated. Specifically in robotics, the data used often does not lie in Euclidean space but in Riemannian Manifolds. Therefore, GaBO can be used to optimize orientations, stiffness or inertia matrices, poses or any of them combined as product of manifolds. This makes many optimization ideas possible and can be extended as an MF setup with a proper lower fidelity.

Bibliography

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008. 10, 14

T. Asfour, L. Kaul, M. Wächter, S. Ottenhaus, P. Weiner, S. Rader, R. Grimm, Y. Zhou, M. Grotz, F. Paus, D. Shingarey, and H. Haubert. ARMAR-6: A collaborative humanoid robot for industrial environments. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 447–454, 2018. 19

T. Asfour, M. Wächter, L. Kaul, S. Rader, P. Weiner, S. Ottenhaus, R. Grimm, Y. Zhou, M. Grotz, and F. Paus. ARMAR-6: A High-Performance Humanoid for Human-Robot Collaboration in Real World Scenarios. *IEEE Robotics & Automation Magazine*, 2019. 20

V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth. Matérn Gaussian processes on Riemannian manifolds. In *NeurIPS Conference on Neural Information Processing Systems*, 2020. 10, 12

M. P. Do Carmo and J. Flaherty Francis. *Riemannian geometry*, volume 6. Springer, 1992. 3, 4

A. Doerr, D. Nguyen-Tuong, A. Marco, S. Schaal, and S. Trimpe. Model-based policy search for automatic tuning of multivariate pid controllers. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5295–5301. IEEE, 2017. vi, viii, 2

D. Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014. 7

P. I. Frazier. A tutorial on Bayesian optimization. *arXiv*, 2018. 5, 11

P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012. 2, 8

N. Jaquier and L. Rozo. High-dimensional Bayesian optimization via nested Riemannian manifolds. *NeurIPS Conference on Neural Information Processing Systems*, 2021. 14

N. Jaquier, L. Rozo, S. Calinon, and M. Bürger. Bayesian optimization meets Riemannian manifolds in robot learning. *CoRL Conference on Robot Learning*, 2019. vi, viii, 1, 2, 4, 9, 10, 12

N. L. G. Jaquier. *Robot skills learning with Riemannian manifolds: Leveraging geometry-awareness in robot learning, optimization and control*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2020. vi, viii, 2, 12

V. Jaquier, N. Borovitskiy, A. Smolensky, A. Terenin, T. Asfour, and L. Rozo. Geometry-aware bayesian optimization in robotics using Riemannian matérn kernels. In *Conference on Robot Learning (CoRL)*, 2021. 2, 12, 14

K. Kandasamy, G. Dasarathy, J. Oliva, J. Schneider, and B. Póczos. Gaussian process optimisation with multi-fidelity evaluations. In *Proceedings of the 30th/International Conference on Advances in Neural Information Processing Systems (NIPS'30)*, 2016. vi, viii, 2, 8, 11, 12, 13, 14, 27

J. M. Lee. *Introduction to Riemannian manifolds*. Springer, 2018. 5

A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. Automatic LQR tuning based on gaussian process global optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 270–277, 2016. vi, viii, 2, 11

A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1563, 2017. vi, viii, 2, 11, 12

J. Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975. vi, viii, 2, 5, 8

C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003a. 7

C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003b. 6

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015. 5

J. Song, Y. Chen, and Y. Yue. A general framework for multi-fidelity Bayesian optimization with gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3158–3167, 2019. vi, viii, 2, 12

N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. 2, 8

K. Welke, N. Vahrenkamp, M. Wächter, M. Kröhnert, and T. Asfour. The armarx framework - supporting high level robot programming through state disclosure. In M. Horbach, editor, *INFORMATIK 2013 – Informatik angepasst an Mensch, Organisation und Umwelt*, pages 2823–2837, Bonn, 2013. Gesellschaft für Informatik e.V. 19, 21

J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020. vi, viii, 2, 11, 12

S. Xiong, P. Z. Qian, and C. J. Wu. Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 55(1):37–46, 2013. 27

T. Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985. 36

K. Yuan, I. Chatzinikolaidis, and Z. Li. Bayesian optimization for whole-body control of high-degree-of-freedom robots through reduction of dimensionality. *IEEE Robotics and Automation Letters*, 4(3):2268–2275, 2019. 11