Incremental Imitation Learning with Estimation of Uncertainty

Inkrementelles Imitationslernen mit Beurteilung der Unsicherheit

Bachelor-Thesis von Claudia Nicole Lölkes aus Hanau

September 2017

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Incremental Imitation Learning with Estimation of Uncertainty
Inkrementelles Imitationslernen mit Beurteilung der Unsicherheit

Vorgelegte Bachelor-Thesis von Claudia Nicole Lölkes aus Hanau

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Guilherme Maeda
3. Gutachten: Marco Ewerton
4. Gutachten: Svenja Stark

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 14. September 2017

(Claudia Nicole Lölkes)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, September 14, 2017

(Claudia Nicole Lölkes)

Abstract

A growing domain to which robotics will eventually evolve is their application in everyday life and routines. As a result, many additional requirements are placed upon robots. In particular, a core requirement when dealing with such applications is the possibility of non-expert users to teach new skills to a robot under a lifelong learning process. To achieve non-expert teaching, imitation learning is a common approach where a human provides demonstrations of a skill to the robot.

This thesis proposes a method for incremental imitation learning using Gaussian process regression. A trajectory in Cartesian space of the endeffector is represented using a linear combination of weighted features, similar to Probabilistic Movement Primitives, and then learned with Gaussian process regression given a context (e.g. a goal position). Gaussian process regression provides the benefits that it has the ability to extrapolate with a few initial demonstrations and offers a principled way to compute the uncertainty. Two major issues that arise when dealing with human-robot interaction are safety and efficiency. In order to guarantee those two requirements, the uncertainty of the model is analyzed. Firstly, the work will show how an active request for a demonstration can prevent the robot from executing undesired and dangerous movements. Such movements occur when the uncertainty to perform a task is too high. Furthermore, active requests can be used to reduce the number of necessary demonstrations if requests are made for contexts with high uncertainty. By using self-exploration, the human-robot interaction for training will decrease even more. This thesis will show how the idea of self-exploration of the context space can be reduced to a supervised learning problem. Lastly, it demonstrates how predicted trajectories can be improved by human refinement in the proposed framework.

The presented methods are evaluated in two experiment setups. Firstly, the methods are analyzed in a toy problem in simulation, where trajectories have to be predicted to a given goal position. Secondly, refinement and self-exploration are evaluated in real robot experiments with a redundant seven degrees of freedom lightweight arm.

Acknowledgments

I would like to thank my supervisors Guilherme Maeda, Marco Ewerton and Svenja Stark for their support, patience and valuable advice. They had always been available and spent a lot of time discussing problems with me. Also, they were the source of many fundamental ideas of my thesis. Guilherme provided advice on the research and algorithm implementations, which helped me formulate and setup the basic experimental framework. A special thank goes to Marco who spent many hours helping me with the experiments with Darias.

Contents

Figures and Tables

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Robots have become widely used. They find application in many processes and domains. However, in most applications nowadays the robots operate isolated from humans and are highly specialized to a specific task. A growing domain to which robotics will eventually evolve is their application in everyday life and routines. Such robots can be used in patient nursing, for support of elderly people or assisting us in our daily lives and at work.
As a result, many additional requirements are placed upon robots. Firstly, lifelong learning that provides the ability to adapt to a new environment and application must be ensured. A major challenge that has to be addressed in order to provide life-long learning is how to teach the robot a new skill without being programmed by an expert each time a situation occurs or a new task is demanded. Better would be to enable a non-expert user to teach the robot. Additionally, the robot should be able to generalize the learned skill to different contexts.

A natural and intuitive way to achieve non-expert teaching is imitation learning. By using imitation learning, a human can give demonstrations in order to tell the robot what to do. This way of teaching is closely related to the way humans and animals learn a new skill defined by a teacher. Figure 1.1 illustrates how a human gives a demonstration to the robot. There are many ways the human can provide the demonstrations, for example, kinesthetic teaching, vision [2] or marker-based [3] tracking of the human or teleoperation [4]. In this thesis, kinesthetic teaching where the teacher guides the robot physically, is applied. An advantage of this approach is that the human is able to refine movements of the robot without giving a completely new demonstration.
However, there are two major issues when dealing with human-robot interaction: safety and efficiency. In order to guarantee those two requirements, this thesis analyzes the uncertainty of the model which the robot has learned.

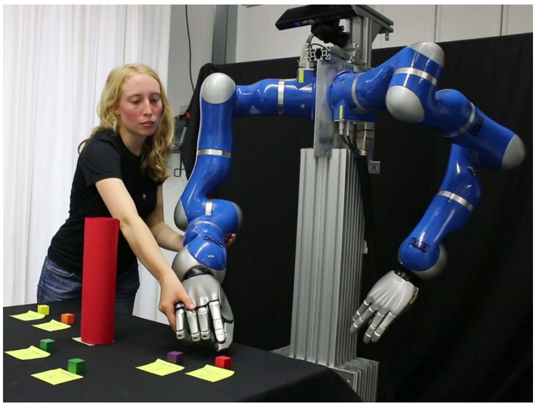


Figure 1.1: Interaction of robot and human: A human teacher demonstrates a new skill to the robot by using kinesthetic teaching.

Safety
First of all, safety is an important requirement when a robot is interacting with humans. The International Electronical Commission defines safety as follows, "Freedom from unacceptable risk of physical injury or of damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment." [5]. When the robot is learning a new skill or is executing a previously learned one, it should not do undesirable or dangerous movements that can harm the human, environment or damage the robot itself. For this purpose, the uncertainty of the learned model is used in this thesis. This way the robot decides by itself when it is able to execute an action and only does so if it is certain enough. Otherwise, it can ask for a new demonstration in order to be able to execute the skill safely.

Efficiency
Secondly, the robot should be taught a new skill with as little human effort as possible. By actively requesting for which context a demonstration should be made, the efficiency can be improved during the teaching phase. In addition, this thesis is going to show how the robot can explore the new skill by itself to improve its knowledge in combination with active requests. By providing refinement, a badly executed skill can be improved easily without the need for a human to provide a whole new demonstration.

1.2 Related Work

Imitation learning in the context of robotics has gained increased attention since its conception in 1980 [6]. In the first years much work was published, mainly addressing offline learning [7] [8] [9]. In order to enable life-long learning and adaption to new tasks, online learning is needed.
Kulić et al. proposed an approach for incremental online learning [10]. In their paper, the human motions were tracked by a motion capture system. The motions were then encoded and mapped to the robot using Hidden Markov Models.

Given that problems with accuracy often occur, Ewerton et al. [11], Ahmadzadeh et al. [1] and Lee and Ott [12] presented approaches where the human can refine robot movements incrementally during the execution of a skill. Since giving a completely new demonstration is not required in this case, incremental refinement provides an efficient option to obtain higher accuracy. Ahmadzadeh et al. modeled a skill as a Generalized Cylinder of multiple demonstrations. The parts needing correction were then refined.
Ewerton et al. aimed at learning motor skills given a context. A trajectory was refined by a refinement loop that adds the weighted error between the executed and refined trajectory to the new trajectory. Moreover, Probabilistic Movement Primitives [13] were used and the dependency of the weights on the context was modeled by a joint probability distribution. This model of the weight-context dependency implies a linear correlation, limiting the complexity of movements. In contrast, this thesis proposes the use of Gaussian processes to provide a more complex weight-context model. Additionally, Gaussian processes perform better in extrapolating with only a few initial demonstrations and give a suitable measure of uncertainty in the extrapolated regions [14].

Schneider and Ertel [15] presented how Gaussian processes can be used for learning from demonstrations and how to deal with the increased computation costs. Also, Farraj et al. [16] and Osa [17] showed how Gaussian processes can be used in imitation learning and for modeling a distribution over trajectories given a task condition.
Maeda et al. [14] extrapolated trajectories by Gaussian processes and encoded them with Dynamical Movement Primitives. The uncertainty given by the Gaussian process was then used to enable active learning. Active learning is a concept, where the algorithm queries the user for desired outputs. This thesis, instead of learning the time-dependent trajectory directly, proposes an approach where the trajectory is first represented using Gaussian basis functions similarly to the Probabilistic Movement Primitive framework [13]. This representation leads to much smoother trajectories and speeds up the learning of the trajectories.
Judah et al. [18] and Shon et al. [19] have also studied active request in the context of imitation learning. While the first focused mainly on giving demonstrations such that only specific states and not full trajectories have to be shown, Shon et al. presented two different approaches on how to choose which demonstration should be given.
One open question that arises when dealing with active request is what metric should be used to trigger a query. Possible approaches are to optimize regarding novelty and the reduction of uncertainty [20] or the confidence in executing an action [21]. Another issue is to automatically adjust the threshold that triggers the query [22].

A robot that is exploring possible policies by itself is a common part of reinforcement learning. In this case, a reward function is needed which is often — especially for a non-robot expert — hard to derive. To solve this problem, imitation learning will be combined with self-exploration by reducing the problem to a supervised learning problem in this thesis. A combination of exploration of the workspace and imitation learning has previously been done. However, this combination has mainly been done for two setups. Firstly, imitation learning was used to bootstrap reinforcement learning algorithms [23] [24]. Here, imitation learning inferred a basic skill which was then refined using reinforcement learning. Secondly, imitation learning was used to learn a reward function from demonstrations [25] [26].

1.3 Outlook

This thesis provides an approach to do imitation learning using Gaussian process regression. In particular, it focuses on analyzing the uncertainty of the learned model in order to provide safety and efficiency for non-robot experts.

Chapter 2 introduces the basic mathematical concepts. First, an introduction to imitation learning and kinesthetic teaching is given. After that, Gaussian process regression and Probabilistic Movement Primitives are described.

Chapter 3 continues by explaining the basic model used to learn trajectories. Afterwards, the concepts of active request, self-exploration and refinement are presented within the used framework.

In **Chapter 4** the experimental setup and results are presented and discussed. Two main experimental setups were used. Firstly, the method was analyzed in a toy problem in which trajectories are drawn to a goal position. Secondly, an experiment with a redundant seven degrees of freedom lightweight arm was set up where the robot has to reach a given position. Each of the approaches of active request, self-exploration and human refinement are then evaluated in the setup.

Finally, in **Chapter 5** the results are discussed and an outlook on future work is given.

2 Foundations

In this section, an introduction is given to the foundations that are used in the model which is presented in this thesis. Section 2.1 is going to explain imitation learning and the method of kinesthetic teaching that is used in the following experiments. In Section 2.2 Gaussian processes are introduced and it is shown how a regression model can be inferred using Gaussian process regression. Lastly, in Section 2.3 Probabilistic Movement Primitives are explained.

2.1 Imitation Learning

Imitation learning by behavioral cloning or learning by demonstration is a field of robot learning where demonstrations given by humans are observed by the robot. A model is then inferred such that the model reproduces the behavior of the demonstrations while generalizing as good as possible. The idea of imitation learning is closely related to the natural behavior of animals and humans and especially for humanoid robots imitation learning provides a simple way to teach new skills to the robot. Usually, imitation learning boils down to a supervised learning problem. In the simplest case a policy $u = \pi(x)$ can be learned from data traces $x_1 \rightarrow u_1 \rightarrow x_2 \rightarrow \ \ldots \ \rightarrow x_n \rightarrow u_n$. In this thesis trajectories given in Cartesian space and represented as a vector $\tau_i = [x_i \ y_i \ z_i]^T$ for each time step in $[1 \ \ T_{max}]$ will be learned as a function $\tau(c)$ of a context c.

2.1.1 Kinesthetic Teaching

Demonstrations can be given in many different ways like teleoperation, vision or marker based tracking or sensuits that are equipped with encoders and accelerometers. In the following experiments, kinesthetic teaching is used to provide the robot with human demonstrations. Kinesthetic learning of humans usually refers to learning by carrying out a physical movement instead of watching demonstrations or reading descriptions of the skill. In the case of robotics, the robot is usually set in a mode where the joints can be moved physically without resistance. A human teacher provides a demonstration by moving the robot in the way the skill should be executed. For example, the robot arm can be moved to a target position to provide a demonstration of a reaching skill.

2.2 Gaussian Processes

In this section, Gaussian processes are introduced and it is shown how a Gaussian process can be fully defined by a mean and a covariance function. Afterwards, it is shown how the regression problem can be solved using Gaussian processes.

A Gaussian process is a stochastic process which can be seen as a generalization of the multivariate Gaussian probability distribution. While a probability distribution provides probabilities for a vector with finite length of random variables, a stochastic process is a collection of possibly infinitely many random variables. Intuitively, an infinitely long vector of random variables can be roughly interpreted as a function $f(x)$ where each entry of the vector represents the value of the function for an input x [27]. Hence, one can think of a Gaussian process as a probability distribution over functions.

Gaussian process. *A time continuous stochastic process $(Z_x)_{x \in X}$ on an arbitrary set of indices X is a Gaussian process if and only if for every finite set of indices $t_1, \ldots, t_n \in T$ the multivariate distribution of $(Z_{x_1}, Z_{x_2}, \ldots Z_{x_n})$ is a n-dimensional Gaussian distribution.*

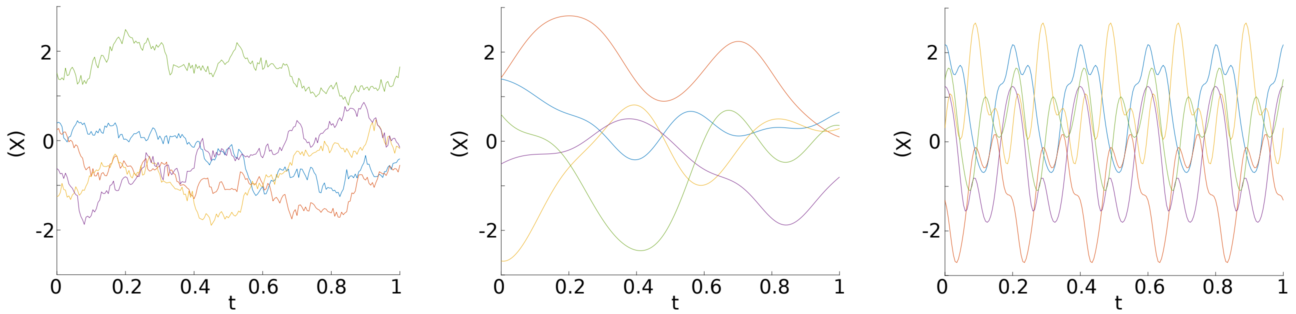
It can be shown that a Gaussian process (Z_x) on X exists for any set X, mean function $\mu : X \rightarrow \mathbb{R}$ and covariance function $k : X \times X \rightarrow \mathbb{R}$ such that:

$$\begin{aligned} \mathbb{E}[Z_x] &= \mu(x) \\ \operatorname{cov}(Z_x, Z_{x'}) &= k(x, x') \quad \forall x, x' \in X. \end{aligned}$$

In the following a Gaussian process will be referred to as:

$$(Z_x) \sim \mathcal{GP}\left(\mu(x), k(x, x')\right).$$

Section 2.2.1 will present how random variables sampled from such a Gaussian process can be used for regression.



(a) **(b)** **(c)**

Figure 2.1: Samples from zero-mean Gaussian processes. For (a) the kernel $\exp\Big(-\sqrt{(x-y)^T(x-y)}\Big)$ is used, in (b) a Gaussian kernel $\exp(-30(x-y)^T(x-y))$ is used and (c) shows samples for the periodic kernel $\exp\Big(-\sin\big(10\pi(x-y)\big)\Big)$

2.2.1 Gaussian Process Regression

Gaussian process regression solves the supervised regression problem. **Supervised learning** is a field of machine learning where a function has to be inferred from labeled training data. Each example of the training data consists of an input value and a corresponding output value. Supervised learning can be divided into regression and classification problems. In **regression** the input data is mapped to continuous output variables.

In this section, it is shown how the regression problem can be solved using Gaussian processes. Rassmussen and Williams [27] provide two insights how Gaussian process models for regression can be interpreted. The following derivation will focus on the first interpretation, the function-space view. From this point of view, one can look at Bayesian linear regression and show that the predictive function

$$f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$$

is in fact a Gaussian process with mean zero and a kernel $k(x,x') = 1/\lambda\ \phi(x)^T\phi(x')$. Secondly, from the weight-space view one can write the predictive function of Bayesian linear regression in terms of a kernel $k(x,x')$ by replacing all inner products $\phi(x)\phi(x')$ using the "Kernel Trick". This approach results in the equation

$$f(x) = \mathbf{k}(\mathbf{x})^T(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}$$

where $\mathbf{k} = (k(x_1,x) \quad \dots \quad k(x_n,x))^T$. In both cases, Gaussian process regression can then be seen as a generalization of Bayesian linear regression with a much richer class of functions.

Lets consider n input values of the model $\{x_1, \ \dots x_l, x_{l+1}, \ \dots x_n\} \in \mathbb{R}^d$ and n corresponding scalar output values $\{y_1, \ \dots y_l, y_{l+1}, \ \dots y_n\} \in \mathbb{R}$, whereof $\mathbf{y} = (y_{l+1}, \ \dots y_n)$ shall be the observed and $\mathbf{f}_* = (y_1, \ \dots y_l)$ the unobserved target values of random variables $(Y_i)_{i\in[1,n]}$. X referes to the $d \times (n-l)$ matrix of aggregated input vectors corresponding to the observed values and X_* to the $d \times l$ matrix of aggregated input vectors corresponding to the unobserved targets.

One can now define n random variables $\mathbf{Z} = (Z_{x_1}, \ \dots Z_{x_n})$ given by a Gaussian process with mean zero

$$(Z_x) \sim \mathcal{GP}\left(0, k(x,x')\right) \ on \ \mathbb{R}^d$$

such that

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} = \mathbf{Z} + \epsilon$$

where $\epsilon \sim \mathcal{N}(0,\sigma^2 I)$ is a random noise variable whose value is independent of (Z_x) [28]. By definition, each finite number of random variables of a Gaussian process is a multivariate Gaussian distribution and the sum of two Gaussians results in a Gaussian distribution again. Consequently one can write

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\Bigg(\mathbf{0}, \begin{bmatrix} K(X,X)+\sigma^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\Bigg)$$

with $k_{ij} = k(x_i, x_j)$ and $K = (k_{ij}) = \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}$.

The unobserved values $\mathbf{f}_*$ are given by the distribution

$$p(\mathbf{f}_*|X,\mathbf{y},X_*) = \mathcal{N}\big(\mu(\mathbf{f}_*),\ cov(\mathbf{f}_*)\big)$$

where

$$\mu(\mathbf{f}_*) = K(X_*,X)\big(K(X,X)+\sigma^2 I\big)^{-1}\mathbf{y}$$
$$\mathrm{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)\big(K(X,X)+\sigma^2 I\big)^{-1}K(X,X_*).$$

$\mu(\mathbf{f}_*)$ is the mean and $\mathrm{cov}(\mathbf{f}_*)$ gives the covariance of the prediction.

Squared Exponential Covariance
In this thesis the squared exponential kernel,

$$k_{SE}(x,x') = \sigma^2 \exp\Big(-\frac{(x-x')^2}{2l^2}\Big)$$

is used which, in Gaussian process regression, corresponds to Bayesian linear regression with infinitely many basis functions [27]. The kernel depends on the length scale l which determines the smoothness of the prediction and the signal variance σ^2 which describes the average distance from the mean.

2.3 Probabilitsic Movement Primitives

Movement Primitives present a way to represent complex movements in robotics. Probabilistic Movement Primitives (ProMPs) were introduced by Paraschos et al. [13] and provide a probability distribution over movements.
Demonstrated trajectories are modeled compactly using a weight vector $\mathbf{w}$ of length N

$$\tau = \Phi w + \epsilon$$

.
The $T \times N$ matrix

$$\Phi = \begin{bmatrix} \phi_1(t_1) & \dots & \phi_N(t_1) \\ \vdots & \ddots & \vdots \\ \phi_1(t_T) & \dots & \phi_N(t_T) \end{bmatrix}$$

represents the basis functions for each time step and $\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is i.i.d. Gaussian noise.
In order to use weights that depend on certain circumstances or contexts and to compute the variance of the trajectory, a distribution $p(w,\theta)$ over the weights and parameters θ is introduced. This distribution can be assumed Gaussian

$$p(w,\theta) = \mathcal{N}(\mu_w, \Sigma_w).$$

such that a closed form solution can be found for $p(\tau,\theta)$ by marginalizing out the weights

$$p(\tau;\theta) = \int p(\tau|w)p(w;\theta)dw = \mathcal{N}(\mu_\tau, \Sigma_\tau)$$

where

$$\mu_\tau = \Phi\mu_w$$
$$\Sigma_\tau = \sigma^2 I + \Phi\Sigma_w\Phi^T$$

μ_τ is the mean of the predicted trajectory and Σ_τ the corresponding covariance.

Linear Ridge Regression

Given time-dependent trajectories the weight representation can be computed using linear ridge regression. The weight vector $\mathbf{w} = [w_1 \ldots w_N]$ can then be computed in closed form

$$\mathbf{w} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T \mathbf{y}.$$

λ is the ridge factor that determines how smooth the predicted function $\Phi\mathbf{w}$ is.

Gaussian Basis Functions

For stroke based movements Paraschos et al. propose Gaussian basis functions of the form

$$b_i(z_t) = \exp\Big(-\frac{(z_t - c_i)^2}{2h}\Big)$$

The basis functions depend on the width h that determines the smoothness of the resulting trajectory representation and the center c_i which changes for each basis function. z_t is a phase variable to decouple the trajectory from the time signal. The basis functions are then normalized with $\phi_i(z_t) = b_i(z_t)/\sum_j b_j(z_t)$.

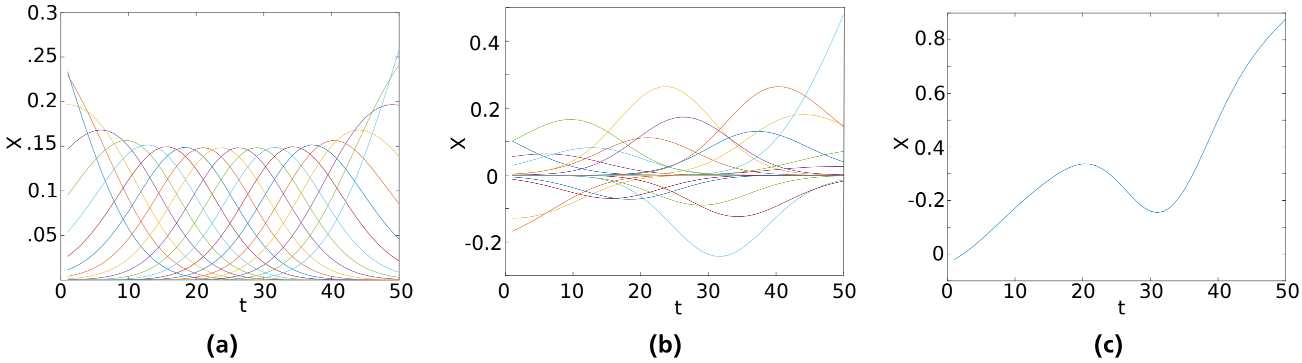


Figure 2.2: (a) shows 20 normalized Gaussian basis functions in the interval [0 50]. In (b) each basis function was multiplied with the corresponding weight which was computed using linear ridge regression. The sum of the weighted Gaussian basis functions results in the trajectory (c).

3 Using Uncertainty of the Model for Safe and Efficient Learning

This chapter presents a method how to learn trajectories and their uncertainty using Gaussian process regression and a weight space representation as in the ProMPs. Depending on a given context Gaussian process regression will predict a trajectory in the representation similar to ProMPs. As a context, the input variable of the Gaussian process will be denoted in this thesis. A context can be for example a goal position, via points or obstacles that have to be avoided. This method will be explained in Section 3.1. Section 3.2 will present how the obtained uncertainty can be used to provide safe and efficient learning using active requests. In Section 3.2 it will be shown how the efficiency regarding human interaction can be improved with self-exploration. Finally, in Section 3.4 it will be demonstrated how human refinement can be realized in this framework.

3.1 Modelling Context-Dependent Trajectories

Gaussian processes have the ability to extrapolate with a few initial demonstrations [14] and provide the uncertainty of the model for each prediction. In this thesis, Gaussian process regression is used to learn how a trajectory should look like for a given context (for example a goal position). If the trajectories are learned using a time-dependent representation directly, the predictions would result in discontinuous trajectories. Therefore, the trajectories are represented using ProMPs. This representation also reduces the number of models that have to be learned using Gaussian process regression, and therefore the computation time, significantly.

The demonstrated and recorded trajectories are assumed to be a vector $\tau_{rec,\ 1:T}$ sampled in T time steps. In the following explanation and in the experiments a Cartesian representation of the trajectory is assumed, such that $\tau_{rec,\ t} = [x_t, y_t, z_t]$ of dimension D. A representation in joint space is also possible and might bring several advantages that are discussed in chapter 5.
The recorded trajectory is then represented using a weight vector,

$$\tau = \Phi w + \epsilon$$

of lenght N. The $T \times N$ matrix

$$\Phi = \begin{bmatrix} \phi_1(t_1) & \dots & \phi_N(t_1) \\ \vdots & \ddots & \vdots \\ \phi_1(t_T) & \dots & \phi_N(t_T) \end{bmatrix}$$

represents the basis functions for each time step and $\epsilon \sim \mathcal{N}(0, \sigma_{ProMP} I)$ is i.i.d. Gaussian noise. The weights can be computed using linear ridge regression,

$$\mathbf{w}_{traj,\ d \in D} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T \tau_{d \in D}$$

for each dimension in D of the recorded trajectory. For each weight of the weight vector $\mathbf{w}_{traj,\ d} = [w_{d,1}, \dots w_{d,N}]$ and for each dimension d, Gaussian process regression is used to predict the scalar weight as a function $f_{d \in D, n \in N}(\mathbf{c}_*) = w^*_{d \in D, n \in N}$ of an unobserved input context $\mathbf{c}_* \in \mathbb{R}^{\mathbf{d}}$. The dimension and number of a weight will be dropped and we will look at a single weight, with $f(\mathbf{c}_*)$ representing a generic dimension and weight number.
Hence, we can construct the training set $\mathcal{D}$ of m given demonstrations $\mathcal{D} = \{(c_i, w_i) \mid i = 1, \dots, m\}$ to train the Gaussian process. C will refer to the matrix of aggregated input vectors of the training set and w to the aggregated weights of the training set. One can now assume the weight values sampled from an underlying Gaussian process and with Gaussian noise $\epsilon_s \sim \mathcal{N}(0, \sigma_s^2 I)$ such that the distribution of the weight for a vector $[\mathbf{w} \ \ f(\mathbf{c}_*)]^T$ of finite length is jointly Gaussian,

$$\begin{bmatrix} \mathbf{w} \\ f(\mathbf{c}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(C,C) + \sigma_s^2 I & K(C, c_*) \\ K(c_*, C) & K(c_*, c_*) \end{bmatrix}\right)$$

with $k_{ij} = k(c_i, c_j)$ and $K = (k_{ij}) = \begin{bmatrix} K(C,C) & K(C,c_*) \\ K(c_*,C) & K(c_*,c_*) \end{bmatrix}$.

As a kernel the squared exponential kernel is used,

$$k_{SE}(c,c') = \sigma_k^2 \exp\Big(-\frac{(c-c')^2}{2l^2}\Big).$$

σ_s^2, σ_k^2 and l are hyperparameters.

The unobserved weight $f(\mathbf{c}_*)$ is given by the distribution

$$p(f(\mathbf{c}_*); C, \mathbf{w}, c_*) = \mathcal{N}\big(\mu_{f(c_*)}, \Sigma_{f(c_*)}\big)$$

with

$$\mu_{f(c_*)} = K(c_*,C)\big(K(C,C) + \sigma_s^2 I\big)^{-1}\mathbf{w}$$
$$\Sigma_{f(c_*)} = K(c_*,c_*) - K(c_*,C)\big(K(C,C) + \sigma_s^2 I\big)^{-1} K(C,c_*).$$

Because we are interested in the mean and the variance of the trajectory, $p(\tau)$ can be found by marginalizing out the predictive weights,

$$p(\tau) = \int p(\tau|f(\mathbf{c}_*))\, p(f(\mathbf{c}_*))\ df(\mathbf{c}_*) = \int \mathcal{N}\big(\tau|\Phi f(\mathbf{c}_*),\ \sigma^2_{ProMP} I\big)\, \mathcal{N}\big(\mu_{f(c_*)},\ \Sigma_{f(c_*)}\big)\ df(\mathbf{c}_*) = \mathcal{N}(\mu_\tau,\ \Sigma_\tau)$$

where

$$\mu_\tau = \Phi\mu_{c_*}$$
$$\Sigma_\tau = \sigma^2_{ProMP} I + \Phi\Sigma_{c_*}\Phi^T$$

with μ_τ the of the predicted trajectory is obtained and Σ_τ is the corresponding covariance which will provide an measurement for the uncertainty of a trajectory execution.

3.1.1 Uncertainty

The uncertainty for a single trajectory is given by $\Sigma_\tau = \sigma^2_{ProMP} I + \Phi\Sigma_{c_*}\Phi^T$. It depends on the uncertainty provided by the Gaussian process and on the hyperparameters σ_s^2, σ_{ProMP} which is the variance of the noise of the ProMPs and the weights and the hyperparameters of the kernel. The uncertainty to reach a context $\mathbf{c}_*$ is computed similar to the proposed heuristic in [14]. To obtain the uncertainty,

$$\text{unc}(\mathbf{c}_*) = \frac{1}{T}\sum_{t=1}^{T}\sqrt{(\sigma^2_{x,t} + \sigma^2_{x,t} + \sigma^2_{x,t})}\ .$$

the sum of the variance entries of Σ_τ, $\sigma^2 = \sum \text{tr}(\Sigma_\tau)$ is accumulated for each dimension and then averaged over all time steps.

3.2 Active Request

In the previous section, a model was obtained that predicts a trajectory for a given context. The model provides not only the trajectory but also the uncertainty to execute this trajectory. This uncertainty can be used to improve the safety and efficiency. One way to use the uncertainty is active request which describes the action of the robot to actively request *when* and for *which context* a new demonstration should be given. There are two scenarios in which active request can be used. Of course, a combination of both is possible.

3.2.1 Improving Safety by Active Request

Trajectories that have a high uncertainty are usually far away from the given demonstrations. The execution of such a trajectory might lead to undesirable movements. Hence, the uncertainty to execute the trajectory can be used to assess if the trajectory should be executed or if the robot should request a new demonstration. An uncertainty trigger unc_{max} can be introduced. Similar to the proposed algorithm in [14], a trajectory for a context c_* is only executed if the robot's uncertainty about the execution unc_{c_*} is below the uncertainty trigger.

$$unc_{c_*} < unc_{max}$$

An algorithm can then be introduced that executes learned trajectories safely and asks for a new demonstration if the uncertainty is too high:

```
Data: $c_*$, $unc_{max}$, $[w_i, c_i]_{i=1:m}$
$[\mu_\tau,\ \Sigma_\tau] \leftarrow$ predict($c_*$) ;
$unc_{c_*} \leftarrow$ uncertainty($\Sigma_\tau$) ;
if $unc_{c*} > unc_{max}$ then
    $[w_{i+1},\ c_{i+1}] \leftarrow$ requestDemonstration($c_*$) ;
    train($[w_i, c_i]_{i=1:(m+1)}$);
    $[\mu_\tau,\ \Sigma_\tau] \leftarrow$ predict($c_*$) ;
end
executeTrajectory($\mu_\tau$) ;
```

Algorithm 1: Safe Execution with Active Request

3.2.2 Improving Efficiency by Active Request

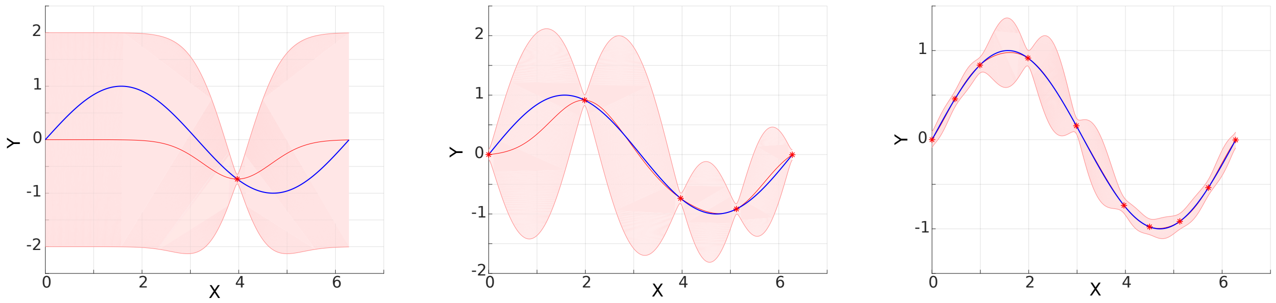
In the training phase the human has to decide which demonstrations should be given. For a human and in particular for a non-expert user this decision is hard to make and might lead to a long training phase until knowledge of the skill was obtained. Instead of letting the human choose for which context c the next demonstration should be given, the robot can make a request for which context c_{req} of the set of possible contexts C the next demonstration should be given. In this thesis the next demonstration will be requested for the context with highest uncertainty. Figure 3.2 and 3.1 show how the uncertainty for a sin-function that is learned using Gaussian process regression is reduced faster if the next training point is given where the uncertainty is highest (Figure 3.1) compared to a random choice of the next training point (Figure 3.2). Furthermore, the Root Mean Squared Error (RMSE) drops faster when choosing the training point with maximum variance (Figure 4.10a).

An algorithm can be introduced in which the robot requests a demonstration where the uncertainty is highest.

```
Data: $C$, $[w_i, c_i]_{i=1:m}$
while $training$ do
    for $c_* \in C \setminus [w_i, c_i]_{i=1:m}$ do
        $[\mu_{\tau,c_*},\ \Sigma_{\tau,c_*}] \leftarrow$ predict($c_*$) ;
        $unc_{c_*} \leftarrow$ uncertainty($\Sigma_\tau$) ;
    end
    $c_{req} \leftarrow$ maxVar($C, unc$) ;
    $[w_{i+1},\ c_{i+1}] \leftarrow$ requestDemonstration($c_{req}$) ;
    $m \leftarrow m + 1$ ;
    train($[w_i, c_i]_{i=1:(m)}$);
end
```

Algorithm 2: Efficient Training with Active Request

An open question is, how to determine how long the model should be trained (i.e. when $training \leftarrow false$). For this purpose, the uncertainty of all contexts in C can be averaged unc_C and compared to a uncertainty trigger unc_{max}. If $unc_C < unc_{max}$ the training stops.

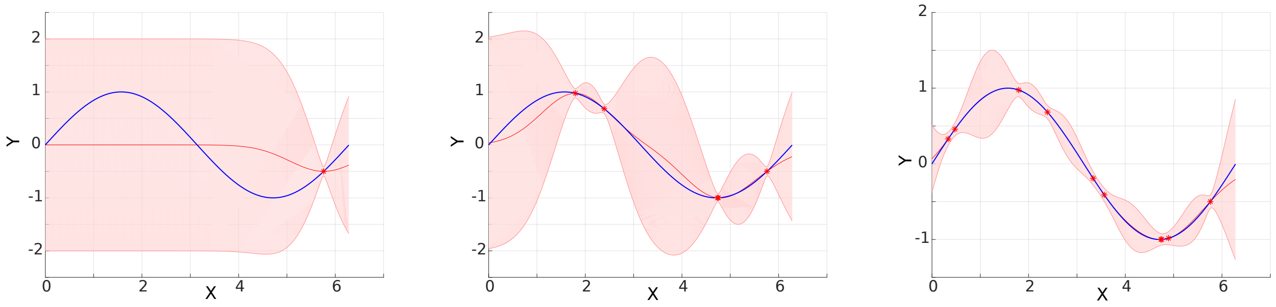


(a) 1 observation

(b) 5 observations

(c) 10 observations

Figure 3.1: Maximum variance choice: a sin-function (blue) is learned using Gaussian process regression. The next demonstration is given where the uncertainty is highest. The mean and variance of the prediction is shown in red.



(a) 1 observation

(b) 5 observations

(c) 10 observations

Figure 3.2: Random choice: a sin-function (blue) is learned using Gaussian process regression. The next demonstration is chosen randomly. The mean and variance of the prediction is shown in red.

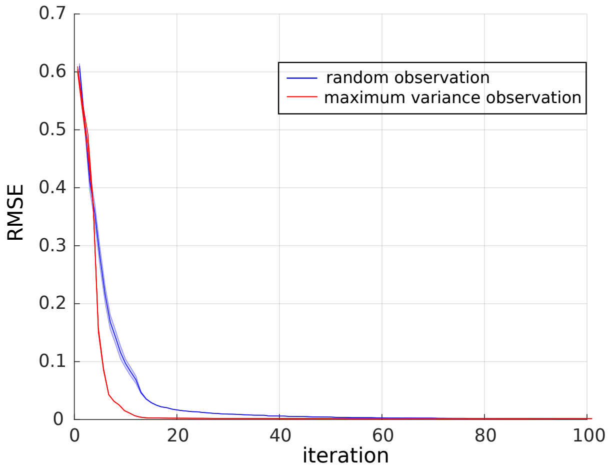


Figure 3.3: The Root Mean Squared Error (RMSE) of learning a sin-function with Gaussian process regression over 50 experiments for 100 iterations. For each iteration a new training point is given. The blue line shows the RMSE for a random choice of the next training point. In red, the RMSE is shown in the case that the next training point is chosen where the variance is highest.

3.3 Self-Exploration

This section proposes how the robot could keep learning without a human who has to give new demonstrations. For this purpose, the property that for a context goal positions are used is exploited in this thesis. Thus, the robot is able to execute an arbitrary trajectory and evaluate the position of its end effector after executing the trajectory even if the end position differs from the goal position c_{goal}. This end position can then be used as a context c_{new} and added to the training data in combination with the executed trajectory. Consequently, self-exploration is reduced to a supervised learning problem.

A disadvantage is that this method can only be used if the context is chosen such that it can be determined when a trajectory is executed.

The question remains which trajectory should be executed to add a new context-trajectory pair to the training data. On one hand, executing trajectories to goal positions with high uncertainty can lead to undesirable movements. One the other hand, the aim is to improve the model as much as possible which would suggest trajectories to positions with the highest uncertainty. Consequently, a balance has to be found between undesired movements and an improvement of the model generalization. One can choose an uncertainty trigger unc_{max}. An algorithm is proposed where goal positions are chosen such that the predicted uncertainty for the goal position is as high as possible but smaller than the uncertainty trigger. It is expected that the model was initialized already with a few initial demonstrations.

```
Data: $C$, $unc_{max}$, $[w_i, c_i]_{i=1:m}$
while $training$ do
    for $c_* \in C \setminus [w_i, c_i]_{i=1:m}$ do
        $[\mu_{\tau,c_*},\ \Sigma_{\tau,c_*}] \leftarrow$ predict($c_*$) ;
        $unc_{c_*} \leftarrow$ uncertainty($\Sigma_\tau$) ;
    end
    $[\mu_{goal}\ c_{goal}] \leftarrow$ maxVarPredictionBelowTrigger($\mu_\tau, unc, unc_{max}$) ;
    $[w_{i+1},\ c_{i+1}] \leftarrow$ executeTrajectory($\mu_{goal}$) ;
    $m \leftarrow m + 1$ ;
    train($[w_i, c_i]_{i=1:(m)}$);
end
```

Algorithm 3: Self-Exploration

3.4 Human Refinement

Sometimes, the trajectory that was executed by the robot does not match exactly the desired trajectory. This discrepancy can occur if the robot is missing knowledge about the environment like obstacles at a place where no demonstrations were given or, for example, if self-exploration was used.

In these cases it is useful if the human can simply refine the executed trajectory incrementally (Figure 3.4) instead of providing a completely new demonstration. In the following algorithm a trajectory that needs refinement is executed a second time and the human can refine the movement during execution. The refined trajectory is then measured during execution and added to the training data. If training data for this context was already provided it is replaced by the new, refined trajectory. Refinement can be done incrementally until the desired trajectory was achieved.

```
Data: $c_*$, $[w_i, c_i]_{i=1:m}$
while $true$ do
    $[\mu_\tau,\ \Sigma_\tau] \leftarrow$ predict($c_*$) ;
    executeTrajectory($\mu_{tau}$) ;
    $refine \leftarrow$ askForRefinement();
    if $refine$ then
        $[w_{i+1},\ c_{i+1}] \leftarrow$ executeTrajectory($\mu_{tau}$) ;
        train($[w_i, c_i]_{i=1:(m+1)}$);
    else
        break;
    end
end
```

Algorithm 4: Self-Exploration

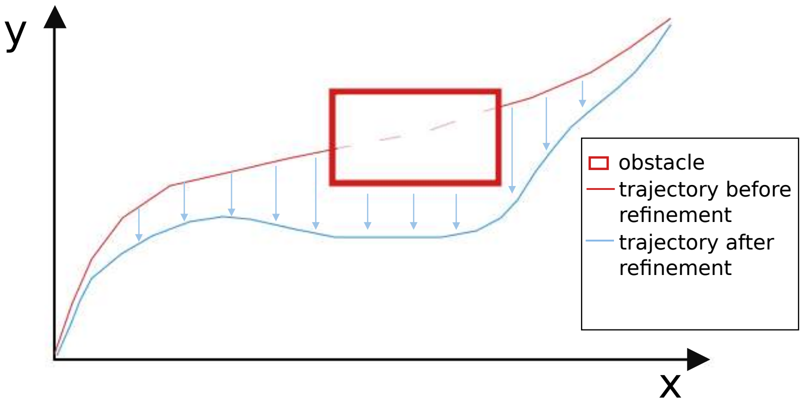


Figure 3.4: If a trajectory does not match the desired trajectory, for example would hit an obstacle (red) refinement of the trajectory changes the new prediction (blue) to the desired trajectory. *(sketch similar to sketch in [1])*

4 Experiments

So far, a method for incremental imitation learning was presented. Several approaches, how the uncertainty of the model can be used to obtain safe and efficient learning, were introduced. In this chapter, the methods are evaluated in two main experimental setups. Firstly, in Section 4.1 a toy problem was implemented where a trajectory has to be drawn in 2D to reach a given goal position. Active request and self-exploration are evaluated in the toy setup. As a second step, experiments were realized with a redundant seven degrees of freedom lightweight arm Darias in Section 4.2. On Darias refinement and self-exploration were tested.

4.1 Experiments in Simulation

In this setup, a path has to be drawn to reach a given goal position. The distribution over weights that model trajectories is initialized with three initial demonstrations given by the user. As a context, two-dimensional goal positions were used. The user then provides N additional goal positions. The learning system tries to reach these positions and orders them with respect to how uncertain it is about the right trajectories to reach them. A new demonstration can then be given to the goal position with the highest uncertainty to increase the confidence in reaching positions. In Section 4.1.2 and 4.1.3, self-exploration is evaluated. At first, contexts are given by the user and the algorithm decides by itself which context it will explore. In Section 4.1.3, a grid of the goal space is then explored with the computer choosing the next goal position by itself.

4.1.1 Active Request

In this section, active request, as described in Section 3.2, is evaluated. It can be shown that when using active request only a few demonstrations are needed until the model is able to predict trajectories for all given contexts.

Experiment Setup

In the experiment, three demonstrations were given to initialize the model. Afterwards, the user provided 10 goal positions. For each goal position, a two-dimensional trajectory is predicted and the uncertainty to reach each context is computed. The contexts are then sorted by their uncertainty (1 indicates the lowest and 10 the highest uncertainty) and the user can provide a new demonstration to the goal position with the highest uncertainty.
The trajectories were encoded in 50 time steps and then parameterized with 5 Gaussian basis functions using linear ridge regression. For the regression, a regularization parameter of 10^{-4} was used to avoid overfitting. The uncertainty to execute a trajectory was considered high if the uncertainty was larger than 0.2 which indicated if a demonstration was still needed or if the model was certain enough to do the prediction without human support.

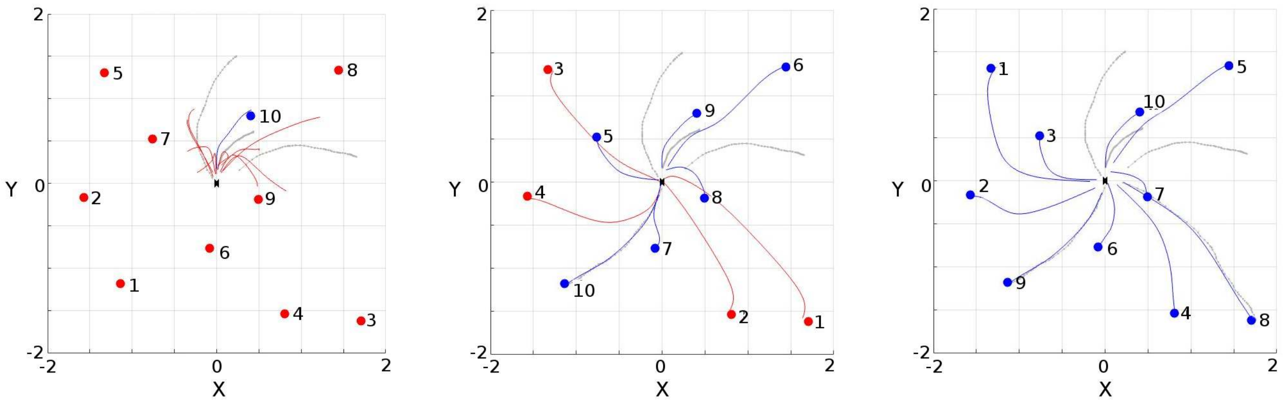
Initial demonstrations	3
Sampling of the trajectory	50 time steps
Number of Gaussian basis functions	5
Regularization Parameter λ	10^{-4}
Number of contexts	10
Uncertainty threshold	0.2

Table 4.1: Values of the parameters that are used to evaluate active request in simulation

In the case of active request, Figure 4.1 illustrates how a total of three iterations showed sufficient to predict trajectories for all contexts. In comparison, when choosing the next demonstration randomly six demonstrations were needed as Figure 4.2 shows. Supplementary to the uncertainty, the accuracy increases faster using active request in contrast to a random choice. The uncertainty provides a way to indicate the user where a new demonstration should be given. Moreover, a possible application is to use the uncertainty to indicate the user if refinement, as tested in Section 4.2.1, is necessary.

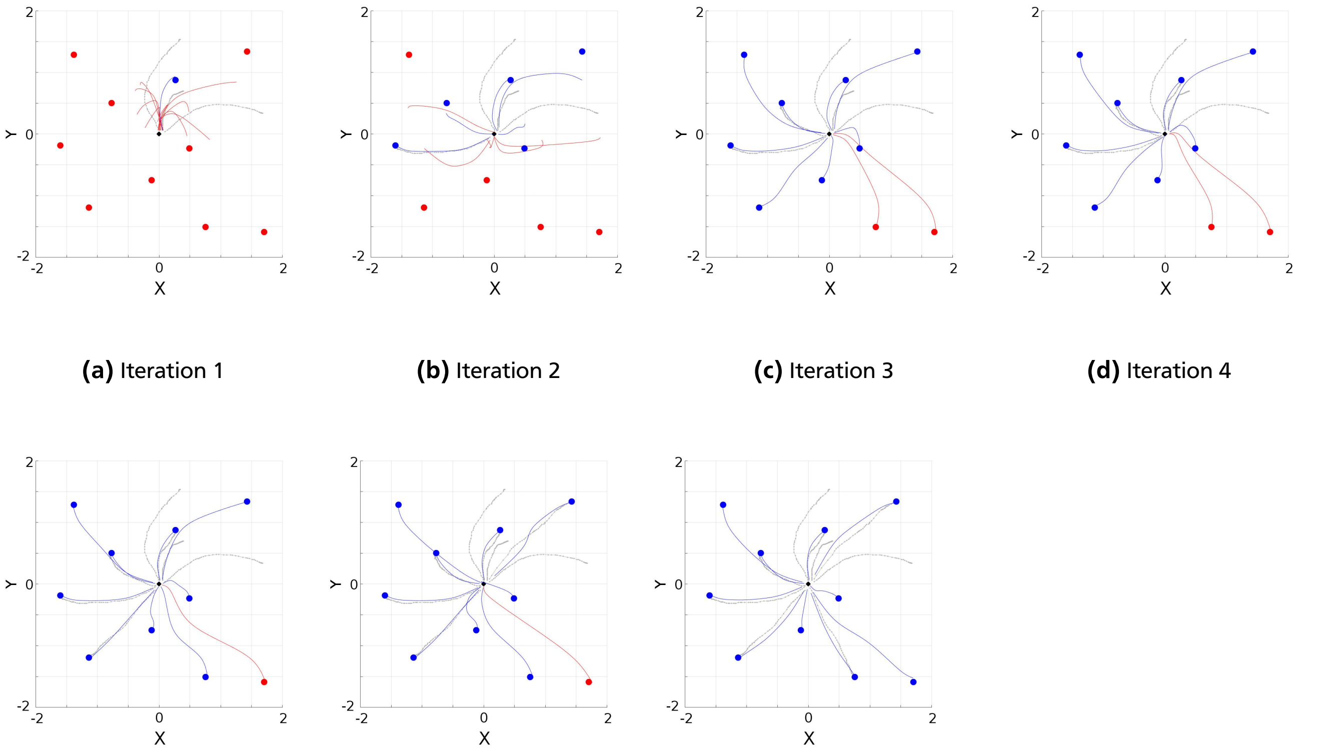
Iteration	1	2	3	4	5	6	7
RMSE active request	2.8054	0.0392	0.0002				
RMSE random choice	1.2735	0.6858	0.0003	0.0002	0.0008	0.0009	0.0009

Table 4.2: The Root Mean Squared Error (RMSE) can be decreased quickly by giving a new demonstration where the uncertainty is highest compared to choosing the next demonstration randomly.



(a) Iteration 1 **(b)** Iteration 2 **(c)** Iteration 3

Figure 4.1: By requesting a demonstration where the uncertainty is highest, only three iterations are needed until the model can predict a trajectory to all goal positions. The contexts and predicted trajectories for which the model is uncertain are shown in red. If the uncertainty is below the uncertainty trigger, the predictions and contexts are shown in blue. The contexts are ordered regarding their uncertainty. The lowest number relates to the highest uncertainty. In the first iteration in (a), after three inital demonstrations (grey) were given, only the trajectory to one context that is close to the initial demonstrations was predicted correctly. In the second (b) and third (c) iteration, a new demonstration to the context with the highest uncertainty was given. After three iterations the model was able to predict the trajectories to all contexts.



(a) Iteration 1 **(b)** Iteration 2 **(c)** Iteration 3 **(d)** Iteration 4

(e) Iteration 5 **(f)** Iteration 6 **(g)** Iteration 7

Figure 4.2: If the choice which demonstration is given is made randomly six iterations are needed until the model is certain about all predictions. In case the model is uncertain about a prediction it is shown in red, certain predictions in blue. The experiment was initialized with three demonstrations (grey) in (a). Each iteration a new context was chosen randomly for which new demonstration was given.

4.1.2 Self-Exploration of Given Contexts

In this section, self-exploration, as depicted in Section 3.3, is analyzed. In the experiment, contexts were given by the user. The algorithm then decided by itself which prediction should be added to the training data. To increase the knowledge, the end position of the predicted trajectory was detected and used as a context for the prediction that was added to the training data.

Experiment Setup

In this experiment three demonstrations were given to initialize the model. Afterwards, the user provided 10 goal positions. For each goal position, a two-dimensional trajectory was predicted and the uncertainty to reach each context was computed. If the uncertainty for a trajectory was below the uncertainty threshold the end position of the trajectory was determined and added, in combination with the trajectory, to the training data. A new demonstration was only given if no trajectory could be added to the training data.

The trajectories were encoded in 50 time steps and then parameterized with 5 Gaussian basis functions using linear ridge regression. For the regression a regularization parameter of 10^{-4} was used to avoid overfitting. The uncertainty to execute a trajectory was considered high if the uncertainty was above 0.2 which indicates if a trajectory was added to the training data.

Initial demonstrations	3
Sampling of the trajectory	50 time steps
Number of Gaussian basis functions	5
Regularization Parameter λ	10^{-4}
Number of contexts	10
Uncertainty threshold	0.2

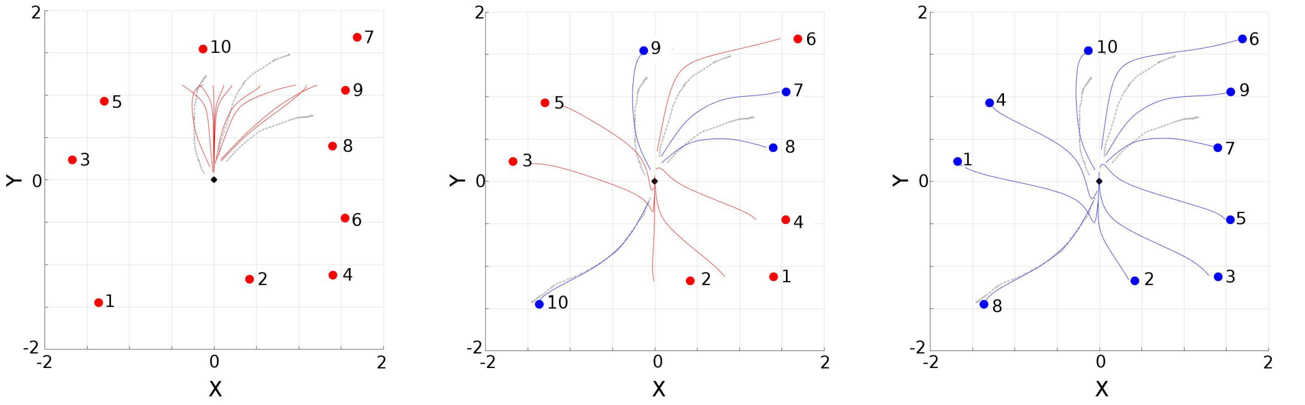
Table 4.3: Values of the parameters that are used to evaluate self-exploration in simulation with contexts given by the user

Experiment Results

We could show that also in the case of self-exploration the algorithm was able to reach all contexts in three iterations with high confidence and with high accuracy. Table 4.4 shows how the Root Mean Squared Error (RMSE) decreases each iteration. After the first three initial demonstrations were given, the algorithm was not certain to reach one of the given contexts. As a consequence, one more demonstration was requested. Figure 4.3 illustrates how the number of uncertain contexts decreased in each iteration.

Iteration	1	2	3
RMSE active request	1.943	0.1163	0.0004

Table 4.4: Using self-exploration the algorithm can increase the accuracy of its predictions.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

Figure 4.3: With self-exploration, the algorithm is able to explore the context space and reduce the uncertainty of the predictions. The contexts are ordered regarding their uncertainty. The lowest number relates to the highest uncertainty. In (a) three initial demonstrations (grey) were given. In the second iteration in (b) one more trajectory had to be demonstrated. In (c) certain predictions (blue) were already added to the training data.

4.1.3 Self-Exploration in a 2D Grid

In this section, self-exploration is analyzed in a two-dimensional grid as described in Section 3.3. Trajectories are predicted to positions on a grid. It can be shown that the algorithm is able to explore the grid incrementally and decrease its uncertainty about reaching positions in the grid.

Experiment Setup

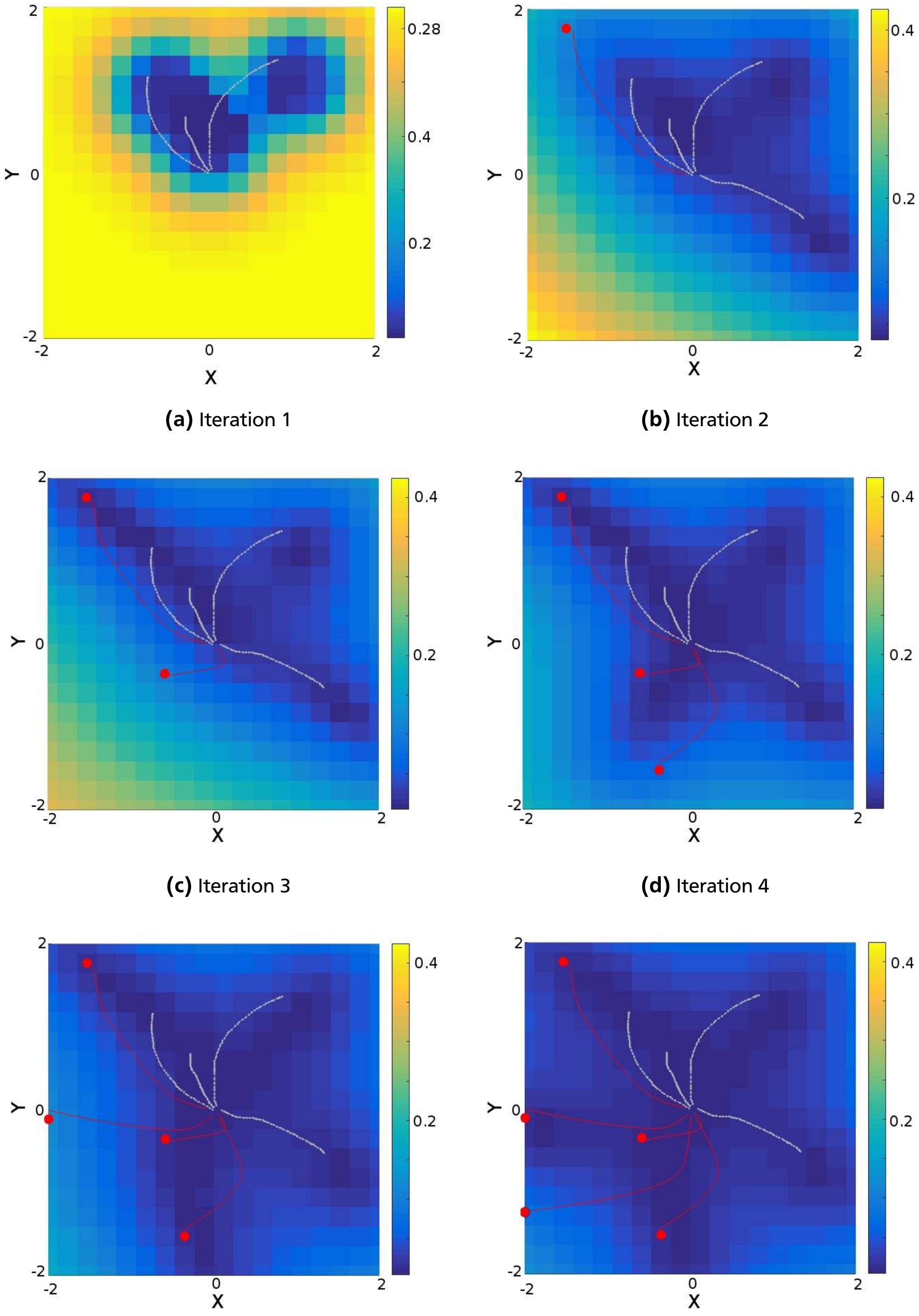
In this experiment, three demonstrations were given to initialize the model. Afterwards, the algorithm explored a 18x18 grid incrementally.
The trajectories were encoded in 50 time steps and then parameterized with 5 Gaussian basis functions using linear ridge regression. For the regression, a regularization parameter of 10^{-4} was used to avoid overfitting. The uncertainty to execute a trajectory was considered high if the uncertainty was larger than 0.1 which indicated if a trajectory is added to the training data.

Initial demonstrations	3
Sampling of the trajectory	50 time steps
Number of Gaussian basis functions	5
Regularization Parameter λ	10^{-4}
Grid size	18x18
Uncertainty threshold	0.1

Table 4.5: Values of the parameters that are used to evaluate self-exploration in a 2D grid in simulation

Experiment Results

After the first three initial demonstrations were given, it was not possible to reach a goal position with high certainty. Therefore, a new demonstration was given. Figure 4.4 shows how the algorithm chooses a goal position afterward and adds the predicted trajectory for the chosen goal position to the training data. Thereby, the model increases its certainty each iteration and expands the areas in which it can certainly predict trajectories. In Section 4.2.2 this setup is transferred to a real robot system.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

Figure 4.4: With each step of self-exploration the uncertainty decreases and the space is extended in which the model is able to predict paths to a goal position.(a)-(f) show the grid which is explored by the algorithm for six Iterations. In (a) three initial demonstrations were given. In the next iterations a new trajectory is drawn and added to the training data.

4.2 Real Robot Experiments on Darias

For the following experiments, the bimanual manipulation platform Darias was used. The robot consists of a torso and two Kuka lightweight robot arms [29]. The end effectors are DLR hands. For the experiments in this thesis, only the right arm was used. The arm has seven degrees of freedom: Three shoulder joints, one elbow joint and three wrist joints. Because the arm is actively compliant, it can be used for kinesthetic teaching and the movements can be refined during execution of a skill. The movement can then be tracked with the torque sensors and joint encoders. To transfer a trajectory to the robot the inverse kinematics were computed using a model of Darias in a high-fidelity simulator (vrep [30]) and then executed using the impedance control of the robot's joints. As context, goal positions were used that were recorded marker based with an Optitrak system at a rate of $90Hz$.

In Section 4.2.1, the movements of the robot are refined using the refinement loop described in Section 3.4. Different situations are evaluated in which the refinement is needed. In Section 4.2.2, the robot explores a grid on a table and the accuracy of the newly learned trajectories is analyzed.

4.2.1 Refinement

In this section, the refinement loop, as depicted in Section 3.4, is evaluated. A reaching skill is performed towards given goal positions. To reach context three a pole has to be avoided which can be shown to the robot using refinement. Afterwards, it is evaluated how well the trajectory for context three can be refined without influencing the trajectories for different goal positions.

Experiment Setup

To evaluate the refinement, a reaching skill was performed. Six goal positions were given on a table. Additionally, a pole had to be avoided when executing a trajectory. The experiment started with three initial demonstrations that were given by a human via kinesthetic teaching. Afterwards, the robot executed trajectories for context one, two and three. Each execution can be refined by the human user if needed, as described in Section 3.4. In case of high uncertainty or if refinement was needed, the movement was executed slowly.

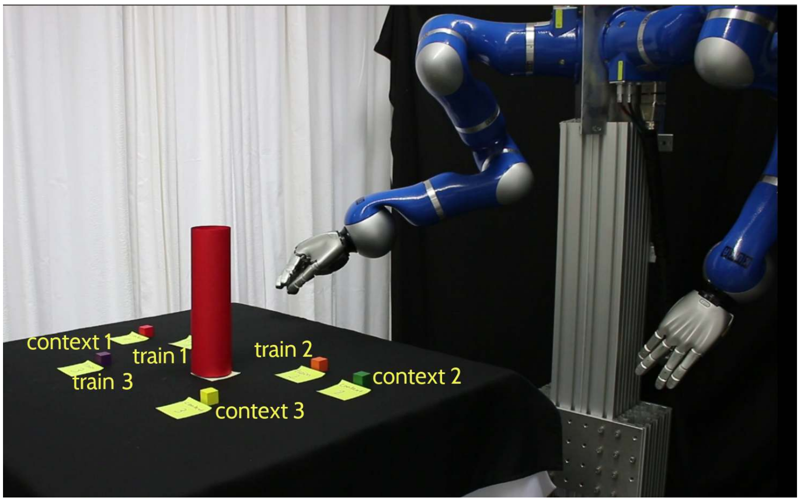


Figure 4.5: In the experiment setup with the bimanual manipulation platform Darias, a reaching skill has to be performed. Three goals are given for training. Then the skill is performed for context one, context two and context three. In order to reach context three a pole has to be avoided.

The three-dimensional trajectories were sampled in 50 time steps and then encoded with 20 Gaussian basis functions using linear ridge regression. For the regression, a regularization parameter of 0.01 was used to avoid overfitting. The uncertainty to execute a trajectory was considered high if the uncertainty was above 0.055 which lead then to a slower execution of the trajectory.

Initial demonstrations	3
Sampling of the trajectory	50 time steps
Number of Gaussian basis functions	20
Regularization Parameter λ	0.01
Uncertainty threshold	0.055

Table 4.6: Values of the parameters that are used for refinement with Darias

Experiment Execution

After the initial demonstrations were given, the robot tried to execute a trajectory towards context one. As shown in Figure 4.7a, context one could be reached while the trajectories for context two and three were not accurate enough at iteration one. In the second step, a trajectory towards context two was performed. In this case, refinement was given in two iterations which lead to more accurate trajectories to context two and three and reduced the uncertainty of all trajectories as shown in Figure 4.7b. The trajectory to context three was executed in iteration six. In this case, the prior knowledge lead to a trajectory, such that the pole was hit. This trajectory was then refined in five steps. Figure 4.7c shows that context three was reached after the refinement. The pole was still slightly touched but did not fall over.

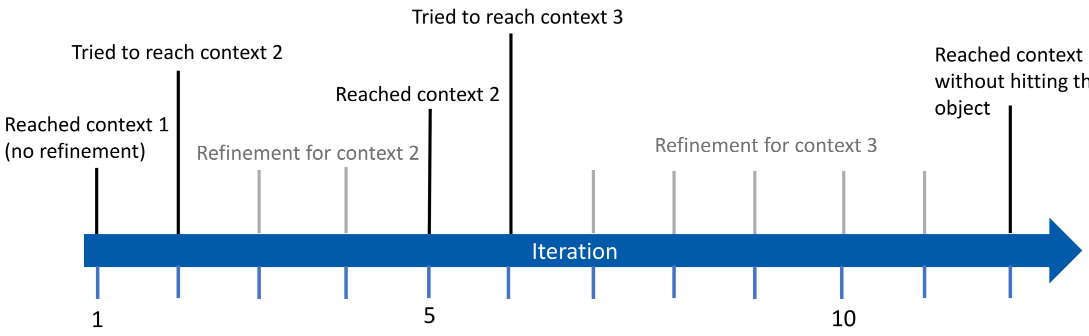
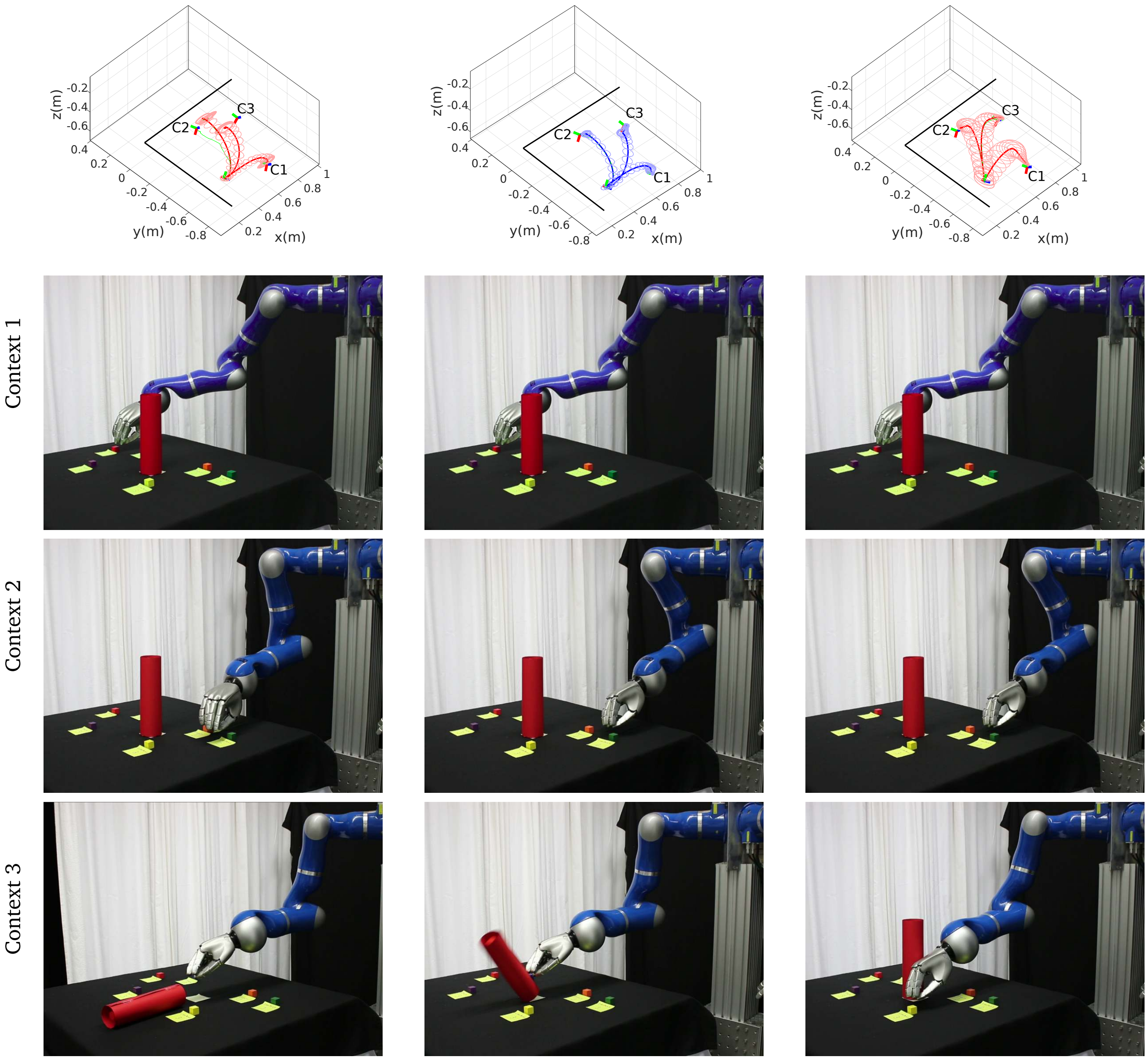


Figure 4.6: Workflow of the experiment: In the first iteration and after the initial demonstrations were given, context one was reached. In iteration 2, a trajectory towards context two was executed. Refinement was then given in two iterations. In iteration six the reaching skill was performed towards context three. In order to avoid the obstacle refinement was given in five iterations for context three.



(a) Predicted Trajectories in Iteration 1 **(b)** Predicted Trajectories in Iteration 5 **(c)** Predicted Trajectories in Iteration 10

Figure 4.7: The figures show the predicted trajectories for three given contexts. (a) illustrates the predictions after the initial demonstrations were given. In (b) the refinement for context 2 was already performed. (c) shows the trajectories after iteration ten when refinement was given for context three.

Experiment Results

The experiment shows that a trajectory can be refined by a human. In Figure 4.8 the robot was not able to reach the goal position at first (orange). While the robot is replaying the skill, the trajectory was refined via kinesthetic correction (red). Darias was able to perform the skill correctly after the refinement was given (blue).
Moreover, the experiment demonstrates how the refinement can be given incrementally if more correction is needed. One problem that occurs is that the refinement of one trajectory influences the prediction of other close contexts. Figure 4.7 and Figure 4.11 illustrate how, as a consequence, the uncertainty of the predictions rises again. The main problem is, that two contexts were considered similar even though the desired trajectories for each of theses contexts had different shapes. In order to solve this problem, the right metric has to be found to measure how similar two contexts are. The squared exponential kernel weights close inputs already stronger compared to inputs that are further away. By tuning the hyperparameters σ_s^2, $sigma_k^2$ and l of the Gaussian process it can be achieved that two contexts are more independent.

Especially, the choice of the kernel or the width l of the kernel should be evaluated in further experiments.

It could also be observed that the endeffector the resulting trajectories of the endeffector corresponded to the desired trajectories that were demonstrated. However, the experiment sometimes also showed convoluted joint movements. The movement for every single joint was computed using the inverse kinematics of a model of the robot in vrep. To solve this problem the computation of the inverse kinematics could be changed. Nevertheless, it would not be possible to refine the the joint movement especially if it is important how the joints move depending on the context. Further experiments should evaluate learning the trajectory in joint space. Even though one major drawback is the higher complexity which has to be dealt with, this approach could solve the observed problem.

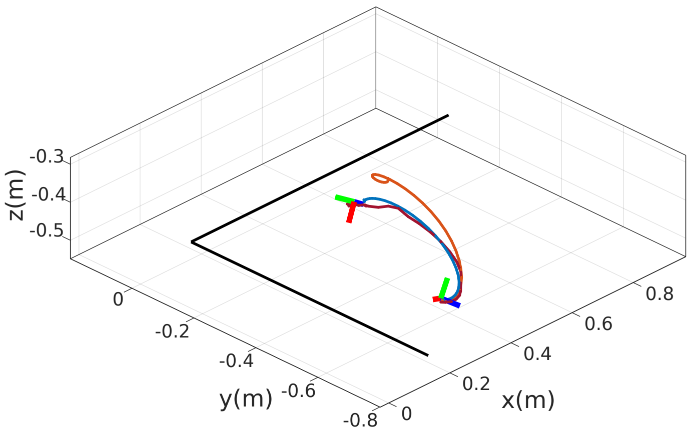


Figure 4.8: A skill can be refined by the human. The plot shows the trajectory before refinement (orange), the human refinement (red) and the trajectory after refinement (blue).

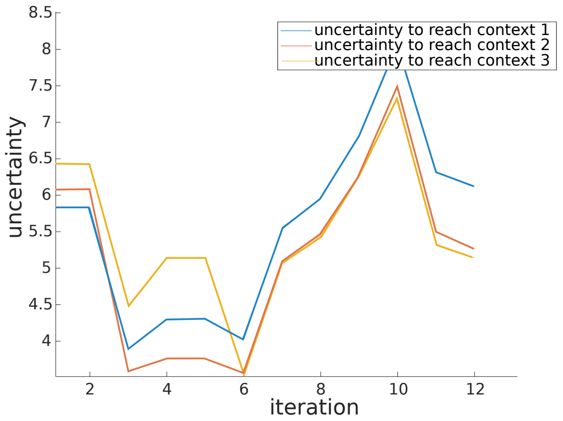
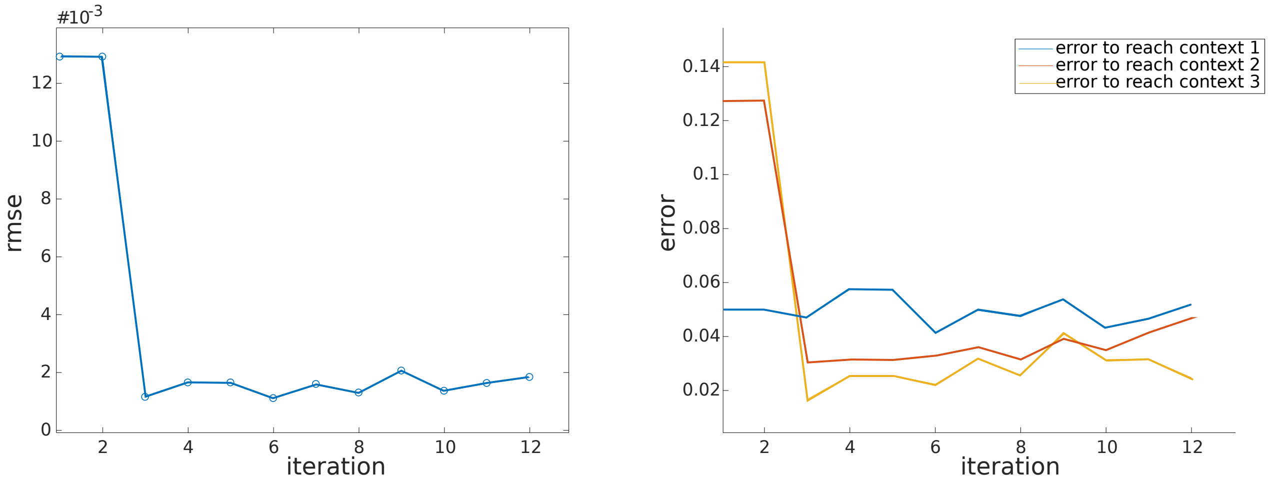


Figure 4.9: In general, adding a new demonstration to the training data reduces the uncertainty of executing a trajectory towards a goal position. However, if a new demonstration differs from the previous ones and is given close to the old training context the uncertainty rises again.

If refinement is given by the human user and the refined trajectory is added to the training data, the RMSE of reaching a goal decreases as illustrated in Figure 4.10a. However, human demonstrations are not perfect and refinement can lead to a slight rise of the RMSE. Figure 4.10b shows the error to reach the goal position for each of the contexts independently. The error of executing a trajectory towards context one is already low in the beginning. For this context, the initial demonstrations provided enough prior knowledge. After refinement for context two was given and the refined trajectory was added to the training data, the model is able to generalize well for context two and three. Nevertheless, refinement for context three was still given because the pole had to be avoided.



(a) RMSE of reaching a goal position over three contexts

(b) Error to reach a goal position for each context

Figure 4.10: More demonstrations and refinement reduce the error of reaching a goal correctly. However, refinement can lead to a small rise due to human imperfection. (a) shows the RMSE of all three contexts. In (b) the error to reach each one of the contexts is plotted independently.

4.2.2 Self-Exploration

In this section, self-exploration is analyzed with the robot Darias as described in Section 3.3. A reaching skill is performed to positions on a grid. It can be shown that the robot is able to explore the grid incrementally by itself, given three initial demonstrations.

Experiment Setup

To evaluate the self-exploration on a real robot, a 10x10 grid was given. The model was initialized with three demonstrations given by the human user. Afterwards, self-exploration of the reaching skill was performed as in the algorithm defined in Section 3.3.

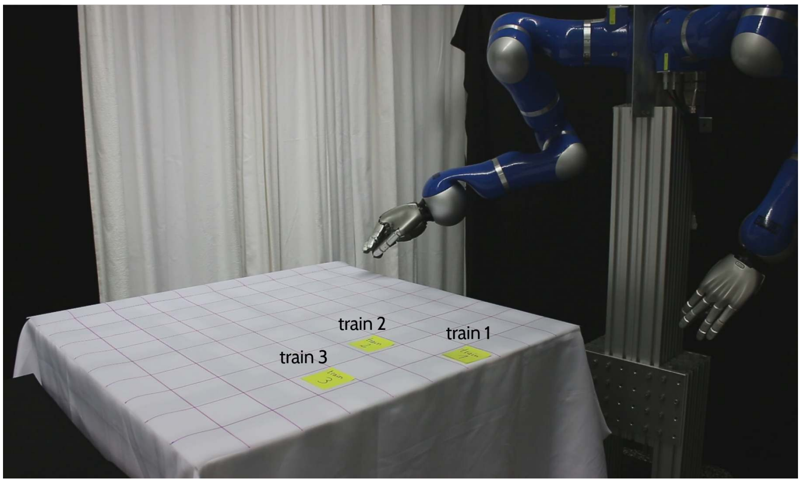


Figure 4.11: In the experiment setup with the bimanual manipulation platform Darias, the robot has to explore a grid. The model is initialized with three demonstrations given via kinesthetic teaching.

The trajectories were sampled in 50 time steps and then encoded with 20 Gaussian basis functions using linear ridge regression. For the regression, a regularization parameter of 0.01 was used to avoid overfitting. The uncertainty trigger that was used to choose the next context for exploration, had a value of 0.065.

Initial demonstrations	3
Sampling of the trajectory	50 time steps
Number of Gaussian basis functions	20
Regularization Parameter λ	0.01
Grid size	10x10
Uncertainty threshold	0.065

Table 4.7: Values of the parameters that are used for self-exploration with Darias

Experiment Execution

After the initial demonstrations were given, the robot had to decide by itself for which context the skill should be performed. The context with the highest uncertainty under the uncertainty threshold is chosen, as in the algorithm defined in Section 3.3. The algorithm evaluates the end position of the trajectory and adds the executed trajectory together with this position to the training data. Figure 4.13 shows how in the first iterations a goal position close to the initial training data is chosen and in later iterations, goals are chosen that are further away.

Experiment Results

The experiment indicates that the robot can increase its knowledge about the context space by self-exploration. Figure 4.13 illustrates how the uncertainty decreases in each iteration and how the area expands in which the robot is certain to execute an action. Additionally to the reduction of the uncertainty, the accuracy increases as Figure 4.12 shows. However, it can also be observed that the model learns to predict trajectories which go first to the area where initial demonstrations were given and then go to the goal position since there is no punishment for generating convoluted trajectories. This motion may lead to undesired trajectories. A solution would be to allow refinement by a human if convoluted trajectories occur. Another solution would be to use reinforcement learning with a cost function that punishes convoluted movements similar to approaches that were proposed in [23] [24].

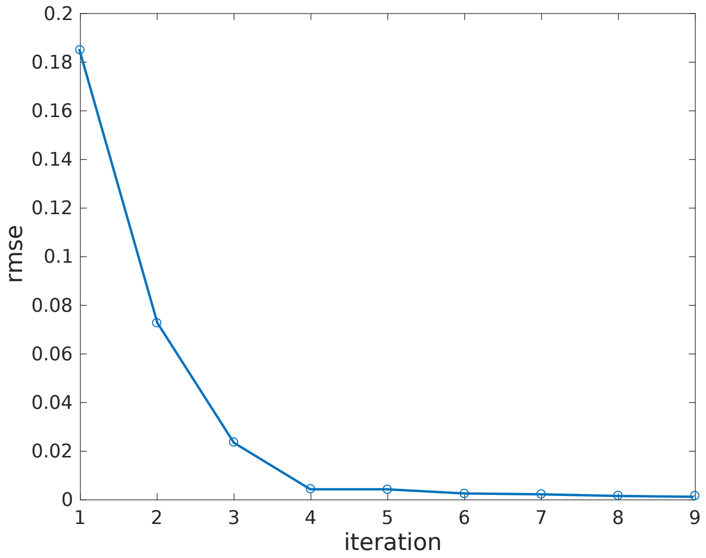
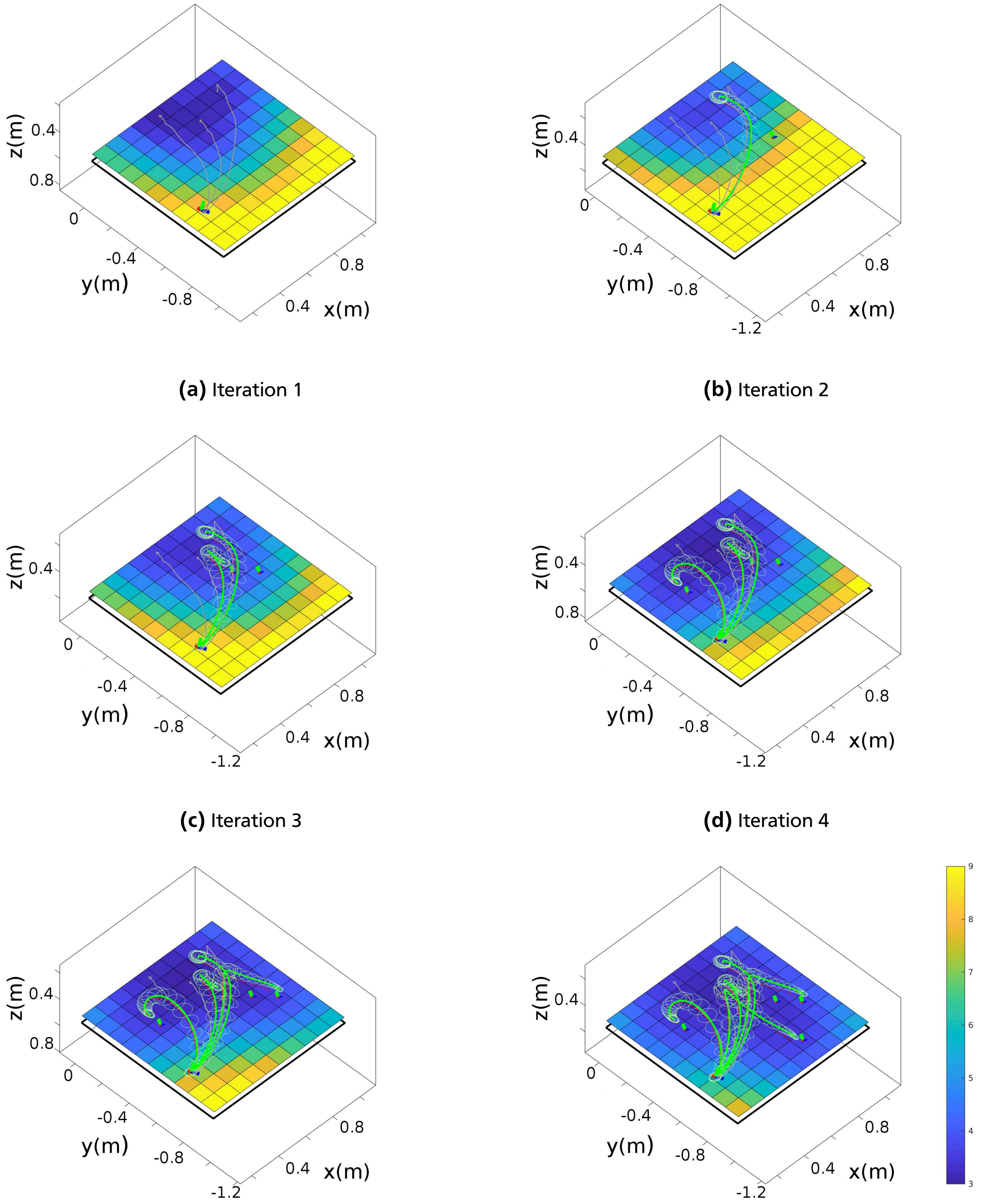


Figure 4.12: By exploring the context space by itself the robot can improve its model of the skill. The plot shows the RMSE at each position in the grid.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

(d) Iteration 4

(e) Iteration 5

(f) Iteration 6

Figure 4.13: With each step of self-exploration the uncertainty decreases and the robot extends the space in which it is able to perform the reaching skill. (a)-(f) show the grid which is explored by the robot for six Iterations. In (a) three initial demonstrations were given. In the next iterations a new trajectory is executed and added to the training data.

5 Conclusion and Future Work

The possibility to teach a new skill for non-expert users and the ability of life-long learning are core requirements of robot applications in everyday life and routines. In particular, safety and efficiency are two major requirements that have to be guaranteed when dealing with human-robot interaction. This thesis presented how incremental imitation learning can be realized using Gaussian process regression. Firstly, a representation of Cartesian trajectories of the endeffector similar to ProMPs was introduced. Gaussian processes offer a measurement of the uncertainty of the model and have the ability to extrapolate with few initial demonstrations in contrast to ProMPs. It was shown how a trajectory can be learned in the presented representation for a given context using Gaussian process regression. A context can be anything on which the desired trajectory depends. In the experiments, goal positions were used as contexts. Different methods were proposed how the uncertainty that was provided by the model can be used to enable safe and efficient learning. Firstly, the work showed how an active request for a demonstration can prevent the robot from executing undesired movements when the uncertainty for performing a task was too high. Furthermore, active requests can be used to reduce the number of necessary demonstrations if requests are made for contexts with high uncertainty. By self-exploration, the human robot interaction for training was decreased even more. The idea of self-exploration could be reduced to a supervised learning problem. In experiments, the robot explored a grid of goal positions by itself. It could be shown that the confidence and knowledge of executing a skill could be increased in the grid. Lastly, we showed how trajectories can be refined in the proposed framework. However, the experiments revealed some problems and open questions which lead to future work that should focus on the following subjects:

Learning in Joint Space

So far, trajectories were learned in Cartesian space of the endeffector. Although learning Cartesian trajectories in task space worked well to predict the desired trajectories of the endeffector, the experiments on the bimanual manipulation platform Darias revealed that the joints sometimes executed unexpected and undesirable movements due to the use of inverse kinematics. Moreover, these movements could not be corrected using refinement because only the endeffector position was considered in the model. Further studies should investigate learning the joint angles instead of the endeffector position over time. Even though one major drawback is the higher complexity which has to be dealt with, it could solve the observed problems.

Learning the Control Model

Modeling a good controller is a hard task in robotics. Instead of learning the trajectories in Cartesian space, a further step could be to learn the control of the joints directly.

Optimizing Active Request

When an active request was made a demonstration for the context with the highest uncertainty was requested in this thesis. With a new demonstration for the requested context, the uncertainty for this context decreases but it is not guaranteed that the reduction of the overall uncertainty is optimized this way. The experiments in the uncertainty grid showed that the reduction of uncertainty for one context can even lead to a rise in uncertainty for other contexts. In general, the goal of active request is to ask for a demonstration such that the information gain with a new demonstration is as big as possible. Different heuristics and methods for active request should be compared and evaluated in further experiments. One should also look at formalizing what, "as much information gained as possible" means and find a method to optimize regarding this measurement. It may prove useful to look into information theory and select the most informative demonstration by maximizing the entropy loss.

Improving Self-Exploration

Self-exploration, as presented in this thesis, showed several limits that should be addressed in further research. One major drawback is that the robot must be able to observe the context after a trajectory was executed in order to reduce the problem to a supervised learning problem which is only possible for a small number of contexts. Furthermore, the experiments showed that convoluted trajectories were generated since there was no punishment for this kind of trajectories. Future work can evaluate if refinement works sufficiently to correct convoluted trajectories or take a closer look into combining reinforcement learning with imitation learning like [23] and [24].

Refinement and Choice for the Right Metric

In the experiments, we showed how trajectories can be refined incrementally in the proposed model. However, one problem that occurred is that refinement of one trajectory influences the movements of the other trajectories if the related contexts are considered similar by the model. The new predictions will then be an interpolation of all demonstrations to close contexts. In the experiments that were evaluated in this thesis, contexts were often considered similar even if the desired trajectories differed. Future work should evaluate which metric should be chosen to achieve the right behavior. The work will result in research of choosing the right values for the hyperparameters and in the choice of the right kernel. When applying the presented ideas to real life applications it should also be considered that situations might also occur in which it is useful to provide a completely new demonstration instead of many refinement iterations.

Evaluation of Different Contexts

So far, only the use of goal positions as contexts was investigated. It would be interesting to combine different contexts in future work. A possible choice of context could be to combine via points with goal positions or to add knowledge of the environment to the context like the positions of obstacles that have to be avoided.

Bibliography

[1] S. R. Ahmadzadeh, M. A. Rana, and S. Chernova, "Generalized cylinders for learning, reproduction, generalization, and refinement of robot skills," in *Robotics: Science and Systems (RSS)*, pp. 1–8, 2017.

[2] M. Do, P. Azad, T. Asfour, and R. Dillmann, "Imitation of human motion on a humanoid robot using non-linear optimization," in *Proceedings of the IEEE/RAS International Conference on Humanoids Robots (HUMANOIDS)*, pp. 545–552, 2008.

[3] N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson, "Adapting human motion for the control of a humanoid robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1390–1397, 2002.

[4] F. B. Farraj, T. Osa, N. Pedemonte, J. Peters, G. Neumann, and P. Giordano, "A learning-based shared control architecture for interactive task execution," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[5] "IEC 61508-1:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems." International Standard, 04 2010. http://www.iec.ch/functionalsafety/explained/.

[6] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[7] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London B: Biological Sciences 358*, p. 537–547, 2003.

[8] V. Krueger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: A review on action recognition and mapping," *Advanced Robotics*, vol. 21, p. 1473–1501, 2007.

[9] C. Breazea and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Science*, vol. 6, no. 11, pp. 481–487, 2002.

[10] D. Kulić, C. Ott, D. Lee, J. Ishikawaand, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing hrough human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[11] M. Ewerton, G. Maeda, G. Kollegger, J. Wiemeyer, and J. Peters, "Incremental imitation learning of context-dependent motor skills," in *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2016.

[12] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, pp. 115–131, Oct 2011.

[13] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*, pp. 2616–2624, Curran Associates, Inc., 2013.

[14] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," *Proceedings of the Conference on Robot Learning (CoRL)*, 2017.

[15] M. Schneider and W. Ertel, "Robot learning by demonstration with local gaussian process regression," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, pp. 255–260, 2010.

[16] F. B. Farraj, T. Osa, N. Pedemonte, J. Peters, G. Neumann, and P. Giordano, "A learning-based shared control architecture for interactive task execution," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[17] T. Osa, N. Sugita, and M. Mitsuishi, "Online trajectory planning in dynamic environments for surgical task automation.," in *Robotics: Science and Systems* (D. Fox, L. E. Kavraki, and H. Kurniawati, eds.), 2014.

[18] K. Judah, A. Fern, and T. G. Dietterich, “Active imitation learning via reduction to I.I.D. active learning,” *Computing Research Repository (CoRR)*, vol. abs/1210.4876, 2012.

[19] A. P. Shon, D. Verma, and R. P. N. Rao, “Active imitation learning,” in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pp. 756–762, 2007.

[20] D. Silver, B. J. A., and A. Stentz, “Active learning from demonstration for robust autonomous navigation,” *Robotics and Automation (ICRA)*, p. 200–207, 2012.

[21] S. Chernova and M. M. Veloso, “Interactive policy learning through confidence-based autonomy,” *Computing Research Repository (CoRR)*, vol. abs/1401.3439, 2014.

[22] S. Chernova and M. Veloso, “Confidence-based policy learning from demonstration using gaussian mixture models,” in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’07, (New York, NY, USA), pp. 233:1–233:8, ACM, 2007.

[23] J. Kober and J. Peters, “Imitation and reinforcement learning - practical algorithms for motor primitive learning in robotics,” *IEEE Robotics and Automation Magazine*, no. 2, pp. 55–62, 2010.

[24] S. Schaal, “Learning from demonstration,” in *Advances in Neural Information Processing Systems 9*, (Cambridge, MA), pp. 1040–1046, MIT Press, 1997.

[25] C. G. Atkeson and S. Schaal, “Robot learning from demonstration,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML ’97, (San Francisco, CA, USA), pp. 12–20, Morgan Kaufmann Publishers Inc., 1997.

[26] K. Subramanian, C. L. Isbell, Jr., and A. L. Thomaz, “Exploration from demonstration for interactive reinforcement learning,” in *Proceedings of the 2016 International Conference on Autonomous Agents &#38; Multiagent Systems*, AAMAS ’16, (Richland, SC), pp. 447–456, International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[27] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[28] “(ML 19.11) GP regression - model and inference.” `https://www.youtube.com/watch?v=JdZr74mtZkU`. Accessed: 2017-09-06.

[29] “Darias: our Bimanual Manipulation Platform .” `http://www.ausy.tu-darmstadt.de/Research/Robots`. Accessed: 2017-09-06.

[30] “vrep.” `http://www.coppeliarobotics.com/`. Accessed: 2017-09-13.