Teaching Robots How to Grasp Like Humans by Humans: An Interactive Approach

Anna Mészáros

Teaching Robots How to Grasp Like Humans by Humans: An Interactive Approach

by

Anna Mészáros

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 23, 2021 at 9:00 AM.

Student number:	5032458	
Project duration:	February 1, 2021 – August 23, 2021	
Thesis committee:	Prof. Dr. -Ing. J. Kober,	TU Delft - Cognitive Robotics Department, supervisor
	Ing. G. Franzese,	TU Delft - Cognitive Robotics Department, supervisor
	Dr. C. H. Corbato,	TU Delft - Cognitive Robotics Department
	Dr. P. Mohajerin Esfahani,	TU Delft - Delft Center for Systems and Control

Acknowledgements

I would like to express my gratitude to everyone from the Cognitive Robotics Department and the AI for Retail Lab for enabling me to delve deeper into the world of robot learning and see the results of the research materialise in the real world.

I wish to thank Jens Kober for his feedback and input both over the course of this project and in the research carried out when I started at the lab. Thanks to his support and that of Luka Peternel and Giovanni Franzese, I was able to take the first steps in my academic career before even starting my thesis. Through the process I learned many things that have broadened my understanding of the academic world and prepared me for the challenges I will face as I continue down the academic path. I would further like to thank Giovanni Franzese for his mentorship during my time at the Cognitive Robotics Department. With his feedback and our discussions on the topic of research, the project provided me with an insightful and pleasant experience.

I would also like to thank the people at the AI for Retail Lab for providing a welcoming working environment. The organised group meetings helped with receiving a fresh perspective on my work and developing ideas further.

This research is funded by the European Research Council Starting Grant TERI "Teaching Robots Interactively", project reference 804907.

Contents

1 Thesis Structure

This thesis is submitted in completion of the Master in Mechanical Engineering with a specialisation in Bio-Robotics. The topic of the thesis covers the learning of dynamic grasping from demonstration and user corrections. The main body of the thesis is provided in paper format. Additional information on the applied methodology can be found in the appendices. The report outline is as follows:

Chapter 2 - the paper resulting from the work of the thesis, detailing the applied methodology, experiments and attained results.

Chapter 3 - provides an investigation into the use of Gaussian Mixture Models and Gaussian Processes for interactive learning.

Chapter 4 - details a potential approach to generalising the proposed framework to different reference frame locations.

Appendix A - provides details on the selection of the orientation representation.

Appendix B - details on the selection of the kernel parameters for the Gaussian Process models.

Appendix C - details on the user interface for providing corrections.

2
Teaching Robots How to Grasp Like Humans: An Interactive Approach

Teaching Robots How to Grasp Like Humans by Humans: An Interactive Approach

Anna Mészáros*, Giovanni Franzese, and Jens Kober

***Abstract*— Grasping objects in a smooth humanlike motion, instead of the more typical pick-and-place approach, includes multiple aspects that need to be performed correctly for a successful grasp. These aspects involve moving the end-effector such that its surface makes and retains contact with the object while also coordinating the movement of the gripper to securely grasp the object. This work investigates how the intricate task of grasping may be learned from humans based on kinesthetic demonstrations. Due to the complexity of the task, these demonstrations are often slow and even slightly flawed, particularly at moments when multiple aspects (i.e. end-effector movement, orientation and gripper width) have to be demonstrated at once. Rather than training a person to provide faster demonstrations, non-expert users are provided with the ability to interactively modify the dynamics of their initial demonstration through teleoperated corrective feedback. This in turn allows them to teach motions outside of their own physical capabilities. In the end, the goal is to obtain a faster but reliable execution of the task. The presented framework learns the desired movement dynamics based on the current Cartesian position with Gaussian Processes, resulting in a reactive, time-invariant policy. Using Gaussian Processes also allows online interactive corrections and active disturbance rejection through epistemic uncertainty minimization. The experimental evaluation of the framework is carried out on a Franka-Emika Panda. Tests were performed to determine i) the framework's effectiveness in successfully learning how to grasp an object quickly, ii) ease of policy correction to environmental changes (i.e. different object shapes and mass), and iii) the framework's usability for non-expert users.**

I. Introduction

More often than not, robots employ a grasping strategy wherein they approach the object, stop and grasp it and only then resume moving. This approach is not necessarily the most natural nor user friendly. In a world where robots are to coexist with humans, generating predictable motions which people can identify and anticipate can help in the acceptance of robots within our environment. Using the standard pick-and-place strategy, the robot may grasp an object and can then decide to move in any possible direction. This might make a person tentative to go close to the robot since they cannot be sure where it will move next. We as humans, on the other hand, tend to grasp things in a single fluent motion where the intent can be anticipated from the trend of the movement. Of course, robots should also be able to complete a task fairly quickly, which in the case of grasping introduces a number of challenges, both from a mechanical point of view as well as a modelling point of view.

Authors are with Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands (e-mail: a.meszaros@student.tudelft.nl, g.franzese, j.kober@tudelft.nl).

*Corresponding Author

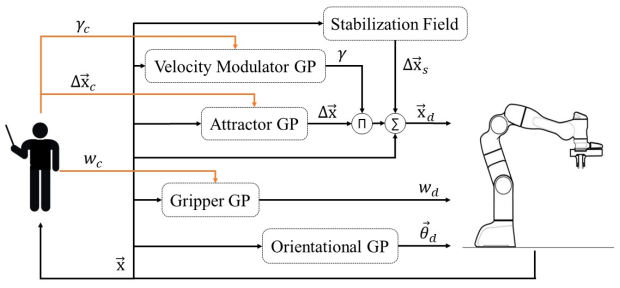


Fig. 1. A schematic representation of the human in the loop giving corrections to the learned motion. The human has a visual feedback of the current motion of the robot and gives corrections with a controller. Δx is the attractor distance in Cartesian space, γ is the scaling factor for altering the magnitude of the attractor, θ is the orientation in Cartesian space and w is the width of the gripper. During policy execution, the human teacher can choose to give corrections or otherwise remove themselves from the loop.

One of the more straightforward ways to teach a robot such an intricate task is through kinesthetic demonstration. Learning from demonstration has become a popular approach for allowing non-expert users to teach robots tasks and thus more easily integrate them into an environment. Yet these provided demonstrations are often slower than what the robot might be able to achieve. This can lead to a reduced efficiency and utility of executing a task with a robot, despite the advantage that robots are able to preform tasks more reliably and for longer periods of time than people. However, if we consider how people learn skills, they first learn to perform a task slowly and correctly, and only once they are confident do they speed up the execution so as not to compromise on the success of the task. This strategy can also be applied to robot learning.

At the same time, an important thing to consider is that the execution of a task often cannot be sped up uniformly. This can either be due to task requirements, such as not spilling water from a glass during sharp turns, or due to limitations of the robot itself. Grasping objects has both limitations. On the side of the task execution, retaining a high velocity when approaching the object can generate high impact forces which can cause the object to bounce away or topple over, potentially damaging the item in question as well as making it impossible to grasp the object on time. On the side of the robot's limitations, certain grippers are not as reactive as our own hands, meaning they might need a longer period to close. In such cases it would be desirable to slow down in the vicinity of the object and speed up after a successful grasp. We as people are able to identify such constraints and adapt accordingly, and through the use of learning from demonstration also transfer this knowledge to the robot.

Of course, when learning to grasp an object in a fluent motion there are bound to be impacts. Depending on the weight and compliance of the object, impacts can generate large errors within the controller which can result in abrupt and potentially dangerous motions. Out of the existing controllers used in literature, impedance controllers are the primary choice in situations involving impacts due to their capability of ensuring a certain degree of compliance. This compliance can be effective for absorbing part of the impact by allowing a margin of error rather than attempting to fully minimise the error to the desired position.

The presented framework continues the study on interactive learning of complex robot motion using Gaussian Process (GP) regression, recently introduced in [1] for learning force tasks. However, this work focuses the attention on interactively learning robot dynamics, orientation and gripper control. The complete behaviour is learned as a function of the robot's position, resulting in a reactive policy. The policy is learned solely from kinesthetic demonstrations and interactive teleoperated corrections. To ensure safe and steady motion along the trajectory, knowledge of the epistemic uncertainty, which can be derived directly from the GP models, is utilised for a simultaneous satisfaction of the desired dynamics and a minimization of the uncertainty. This provides an elegant solution to covariate shift, avoiding dangerous extrapolation in particular when performing high speed motions.

II. Background and Related Work

A. Robot motion with impacts

When executing high-speed manipulation tasks which involve establishing contact with an object, it is important to consider the behaviour around the moment of impact. A reoccurring approach observed in existing works consists of adapting the relative velocity in order to mitigate the effects of the impact [2], [3]. Another strategy which has been employed to absorb impacts particularly in catching tasks, involves utilising a follow-through behaviour which continues to track the predicted path of an object even after interception [3], [4]. A follow-through behaviour, however, is not applicable to stationary objects. A similar behaviour can nevertheless be obtained by employing a variable impedance controller with low stiffness. Indications of the benefits of impedance control for absorbing impacts could be observed in other contact tasks such as quadrupedal trotting [5], and learning a hopping movement on a floating base system [6].

Unlike most standard controllers, impedance control has the capability of incorporating a compliant behaviour into a provided attractor [7]. This is achieved by modelling the attractor in the form of a mass-spring-damper system. Depending on the defined system properties for inertia $\mathbf{\Lambda}$, damping $\boldsymbol{D}$ and stiffness $\boldsymbol{K}_s$, the dynamics of the end-effector in the case of *translation impedance* are given through the following relation

$$\mathbf{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} = \boldsymbol{K}_s \Delta \boldsymbol{x} - \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{f}_{\text{ext}}, \tag{1}$$

with f_{ext} representing the force at the end-effector where the gravitational and Coriolis terms have been pre-compensated. Considering this relation, it is possible to induce compliant behaviour which allows a softer interaction with the object.

An impedance controller, however, is unable to mitigate the initial impact force irrespective of the set stiffness as demonstrated in [8]. This is because the main contribution to the impact force is the velocity of the impacting objects. The results presented in [8] do, however, strengthen the observation that increased compliance provided by impedance is beneficial for absorbing the forces post-impact. While matching the velocity of an object likely achieves the best reduction of impact force, such an approach may not be optimal if considering the speed of the trajectory execution. This is especially true for the case of a static object, wherein matching velocities would effectively bring the robot to a stand-still prior to the grasping action. The best approach, therefore, is to learn a velocity profile. Doing so makes it possible to handle impacts at non-zero velocity by ensuring moderate impact forces through adequately low velocities.

B. Learning Dynamic Movements

Teaching a complex trajectory to a robot with the desired final dynamics can be challenging. If the transfer of knowledge is to be executed correctly, often the best strategy is to show the desired behaviour slowly in order to reduce mistakes. However, the speed at which the demonstration was carried out may not be the desired speed of execution. Being able to adapt the learned velocity with ease plays a key role in speeding up the overall execution of the demonstrated trajectory while also considering that the movement dynamics may require different degrees of adaptation at different points of the trajectory; for example partly slowing down prior to the moment of interception. Different works explore speed adaptation during trajectory execution using different function approximators. One approach involves altering the phase rate of probabilistic movement primitives (ProMPs) [9], [10], whereas another sees the use of a modified version of DMPs in which speed is altered through an additional phase-dependent temporal scaling factor [2], [11]. The ease of adapting a trajectory through corrective feedback is however hampered in this case due to the phase variable and its indirect dependence on time. If such an approach is to be used it would be necessary to update the phase accordingly during the interaction with the user to ensure that the trajectory is carried out as expected.

An alternative to this is using Gaussian-based regression such as Gaussian Mixture Regression (GMR) or Gaussian Processes (GP). Gaussian Mixture Models (GMMs), in particular, have been utilised in works investigating the catching [4], [12] and reaching [13] of objects, which is closely related to grasping objects quickly. These approaches utilise Gaussian Mixture Models in order to model the dynamical system and GMR to extract the outputs of the learned system. Velocity adaptation is carried out in accordance to a separate scaling factor which is determined online. Rather than mathematically formulating the scaling factor for the velocity as above, Locally Modulated Dynamical Systems [14] learn this factor with the help of user input. This input is then used to learn the scaling factor through a Gaussian Process. It is, however, worth noting that in the experiments the learned modulation factors were used to alter the shape of the trajectory, rather than speed up the execution.

In terms of grasping objects while in movement, existing strategies employ motion planning such as in [15]. Such strategies, however, are time dependent and require re-planning of the desired motion in the event of changes to the environment. Furthermore, for motion planning, environment and task constraints have to be mathematically formalised in order to be incorporated into the planning process.

Our proposed framework aims to enable the teaching of this highly dynamic task through learning from demonstration in order to allow even non-expert users to program the robot's motions. Furthermore, the learned policies create a reactive control, allowing real-time adaptations of the behaviour depending on the robot's current location. To our knowledge, as of yet, there exists no solution addressing these challenges.

III. Methodology

The goal of this framework is to enable a user to teach the robot the desired motion through demonstration and correction, see Alg. 1. The robot is learning the desired minimum uncertainty dynamical system on the end-effector, formalized in Sec. III-A and the dynamics of the gripper orientation and width as a direct mapping to the current robot position, formalized in Sec. III-B. The main aim is to show that it is possible to learn a policy and later correct the velocity. All of these aspects are modelled with Gaussian Processes allowing interactive corrections of the dynamics and constraints online, see Sec. III-C.

A. Minimum Uncertainty Dynamical System

A simple dynamical system can be described by

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \tag{2}$$

where $\boldsymbol{x}$ is the robot state and f regulates the transition of the robot state. This type of formulation would fit perfectly in a velocity controller, however, due to the necessity of dealing with impacts, for which an impedance controller is more suitable, we can reformulate the equation as

$$\boldsymbol{x}_d = \boldsymbol{x} + \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x} + \Delta\boldsymbol{x}(\boldsymbol{x}) \tag{3}$$

where $\boldsymbol{x}_d$ is the desired attractor position. The dynamical system can be seen as an external (and slower) control loop where the attractor position is updated as a function of the robot position while the inner (and faster) control loop simulates the dynamics of a critically damped second order dynamical system. In order to allow to modify the magnitude of the attractor distance proportionally in all directions, a scalar factor is learned as a function of the position, resulting in

$$\boldsymbol{x}_d = \boldsymbol{x} + f(\boldsymbol{x}) = \boldsymbol{x} + \gamma(\boldsymbol{x})\Delta\boldsymbol{x}(\boldsymbol{x}) \tag{4}$$

where $\Delta\boldsymbol{x}$ is fitted with a Gaussian Process using the data of a kinesthetic demonstration and user-provided corrections. The scaling factor γ is also modelled using a GP but is initialized to a constant value for all points of the trajectory, which can later be adapted through corrections. Nonetheless, since both $\Delta\boldsymbol{x}$ and γ are affecting the attractor, the correlations between the known datapoints and therefore the parameters of the GPs can be taken as equivalent.

Briefly on Gaussian Proccesses; a GP is a non-parametric regression method that provides the means for inferring the expected output and the epistemic uncertainty. The mean and variance of the process are denoted as

$$\mu(\boldsymbol{x}) = \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x})^\top (\boldsymbol{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \tag{5}$$

$$\Sigma = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x})^\top (\boldsymbol{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x}), \tag{6}$$

where k is the variance of the single evaluation point $\boldsymbol{x}$, $\boldsymbol{k}_*$ is the covariance between $\boldsymbol{x}$ and the training inputs $\boldsymbol{\xi}$, $\boldsymbol{K}$ is the covariance matrix of the training inputs, σ_n^2 is the variance of the Gaussian noise of the training points, and $\boldsymbol{y}$ denotes the training outputs [16]. Both k and $\boldsymbol{k}_*$ as well as $\boldsymbol{K}$ are a function of the kernel and its hyper-parameters used to incorporate prior knowledge in the process.

Something to consider when learning a dynamical system is that the next robot position is a function of the learned desired transition and the external disturbances. This may lead the robot in a region where its policy is not confident anymore. Depending on where this might occur, the robot may not be able to successfully grasp the object or bring it to its goal and successfully execute its motion. To avoid this from happening, the dynamical system was augmented with another component that brings the robot towards regions of low uncertainty. From a practical point of view this results in adding another attractor field that is proportional to the gradient of the variance manifold [1] according to

$$\Delta\boldsymbol{x}_s(\boldsymbol{x}) = -\alpha\nabla\Sigma = \alpha \left(2\boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \frac{\partial \boldsymbol{k}_*}{\partial \boldsymbol{x}} \right) \tag{7}$$

where $\boldsymbol{x}$ is the evaluation point, and α is an automatically modulated constant according to a maximum allowed attractor distance, which ensures that $\Delta\boldsymbol{x}_s$ is never higher than a set threshold. Thus, the Minimum Uncertainty Dynamical System (MUDS) can be summarized as

$$\boldsymbol{x}_d = \boldsymbol{x} + \gamma(\boldsymbol{x})\Delta\boldsymbol{x}(\boldsymbol{x}) - \alpha\boldsymbol{\nabla}\Sigma(\boldsymbol{x}) \tag{8}$$

This approach is effective due to two factors. The first is that as the evaluation point moves further from the known region, the values of the correlation vector k_* begin to tend to zero. In turn, the predictions begin to vanish towards the mean of the independent Process, which for $\Delta\boldsymbol{x}$ is zero. Simultaneously, the variance begins to increase and gradually activate the uncertainty minimisation. The uncertainty minimisation field is thus able to redirect the robot towards the region of low uncertainty.

Algorithm 1: The basic framework.

```
1  a) Kinesthetic Demonstration(s)
2  while Trajectory Recording do
3      Receive($\boldsymbol{x}_t$, $\sin\boldsymbol{\theta}_d(\boldsymbol{x}_t)$, $\cos\boldsymbol{\theta}_d(\boldsymbol{x}_t)$, $w_d(\boldsymbol{x}_t)$)
4      $\Delta\boldsymbol{x}_d(\boldsymbol{x}_{t-1}) = (\boldsymbol{x}_t - \boldsymbol{x}_{t-1})$
5  end
6  Train(GPs)
7  b) Interactive Corrections
   Data: $\Delta\boldsymbol{x}_d$, $\gamma_d$, $\sin\boldsymbol{\theta}_d$, $\cos\boldsymbol{\theta}_d$, $w_d$
8  while Control Active do
9      Receive($\boldsymbol{x}$)
10     if Received Human feedback $\Delta\boldsymbol{x}_c$, $\gamma_c$, $w_c$ then
11         Correct($\Delta\boldsymbol{x}_c \rightarrow \Delta\boldsymbol{x}_d$, $\gamma_c \rightarrow \gamma_d$, $w_c \rightarrow w_d$)
12     end
13     $[\Delta\boldsymbol{x}, \Sigma]$ = $\mathrm{GP}_{\Delta\boldsymbol{x}}(\boldsymbol{x})$
14     $\gamma$= $\mathrm{GP}_{\gamma}(\boldsymbol{x})$
15     $w_d = \mathrm{GP}^{\delta}_{w}(\boldsymbol{x})$
16     $[\sin\boldsymbol{\theta}, \cos\boldsymbol{\theta}] = \mathrm{GP}^{\delta}_{\theta}(\boldsymbol{x})$
17     $\boldsymbol{\theta}_d = \arctan 2(\sin\boldsymbol{\theta}, \cos\boldsymbol{\theta})$
18     $\boldsymbol{x}_d = \boldsymbol{x} + \gamma\Delta\boldsymbol{x} - \alpha\nabla\Sigma$
19     Send($\boldsymbol{x}_d, \boldsymbol{\theta}_d, w_d$)
20 end
```

B. Learning Minimum Uncertainty Mappings

When learning a complex task like grasping, the dynamics of the end-effector position have to be augmented with the dynamics of the gripper orientation and width. Because the dynamics are always coupled, we decided to learn a mapping between the robot position and the controlled variable with a Gaussian Process. However, if the predictions are done based on the current position, when outside of the region of certainty, the robot would output the mean of an independent Process (i.e. zero radians for the orientation along all three axes and maximum gripper width) which could lead to an undesirable generalization, e.g. tilting or dropping objects. In order to solve this problem, what we propose is a new Gaussian Process prediction according to

$$\mu(\boldsymbol{x}) = \boldsymbol{k}^{\boldsymbol{\delta}}_{*}(\boldsymbol{\xi}, \boldsymbol{x})^{\top}\boldsymbol{K}(\boldsymbol{\xi}, \boldsymbol{\xi})(\boldsymbol{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{y} \quad (9)$$

where $\boldsymbol{k}^{\boldsymbol{\delta}}_{*}$ is the correlation vector filtered with a Kronecker filter. Each element is computed according to

$$k_i^{\delta} = \begin{cases} 1, & \text{where } \underset{i}{\operatorname{argmax}}(k(\boldsymbol{\xi}_i, \boldsymbol{x})) \\ 0, & \text{elsewhere} \end{cases} \quad (10)$$

where $\boldsymbol{\xi}_i$ is the i-th element of the database. This new prediction can be interpreted as a "mental" simulation towards the direction of minimum uncertainty. The aim is to explicitly avoid extrapolating the data outside the original demonstrated data while still using the property of smooth regressor of the Gaussian Process. When the evaluation of the Gaussian process is performed with this extrapolation-free rule, we denote them with the superscript δ, i.e. GP^{δ}.

In order to fit the desired angles with a regressor, it was necessary to have a smooth and continuous representation of the angles. To this end we learned both the $\sin$ and $\cos$ transformation of the orientation $\boldsymbol{\theta}$ and we converted the predictions back when in autonomous control (l. 19 Alg. 1).

C. Interactive Policy Correction

The main idea of the proposed framework is to enable the teacher to locally modify and speed up their initial demonstration. Since adding new points to the database would expand the known region it could drastically alter the shape of the trajectory, particularly when providing corrections to the attractor direction after having provided corrections to the velocity. For this reason, only the outputs of the database and in turn the policy could be altered through interactive corrections. This choice has been made under the assumption that in the case of a high velocity task execution, one does not wish to stray from the demonstrated region. While it does limit the scope within which the user can correct a previously shown demonstration, allowing the user to only correct the existing data points and not add new ones also guarantees control at higher frequencies, which can be important for the proper execution of fast-paced tasks, since no matrix inversion has to be performed for Eq. 5.

The evaluation of the process kernel allows the corrective input to be smoothly spread to surrounding data points in accordance to their correlation. The update rule was thus chosen as

$$\boldsymbol{y}^d = \boldsymbol{y} + \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x})\epsilon_{\mu} \quad (11)$$

where ϵ_{μ} is the correction provided at $\boldsymbol{x}$. This rule was applied for correcting the attractor distance, velocity modulation factor and the width of the gripper prongs. It has previously been shown that spreading the corrections on the database is more user-friendly, as well as time and data efficient [1] than a simpler data aggregation [17], since otherwise the GP model would essentially average between the different outputs for a given input, leading to a slow learning.

When users provide corrections to the dynamics, they can only provide information on the speed and direction of movement. This can, however, lead to abrupt changes in the acceleration. Therefore, at the end of every correction round, if the datasets for the attractor or the scaling factor have been altered they are passed through a Savitzky–Golay filter [18]. The filter fits a polynomial of a defined order to a set of points within a chosen window. The polynomial fit is then used to estimate the point in the middle of the window, after which the window is shifted by one position and the procedure is carried out again. The result is a smoothed dataset, ensuring smooth motion when the model is refitted. This operation is only valid if no new points are appended to the database. Otherwise, new points could be smoothed with non-neighbouring points, potentially changing the robot's behaviour outside of the user's intentions.

As a final remark, it is worth underlining that the capability of correcting the orientation after the demonstration was not enabled due to the limitations of the teleoperation interface, not due to any limitations surrounding the algorithm itself.

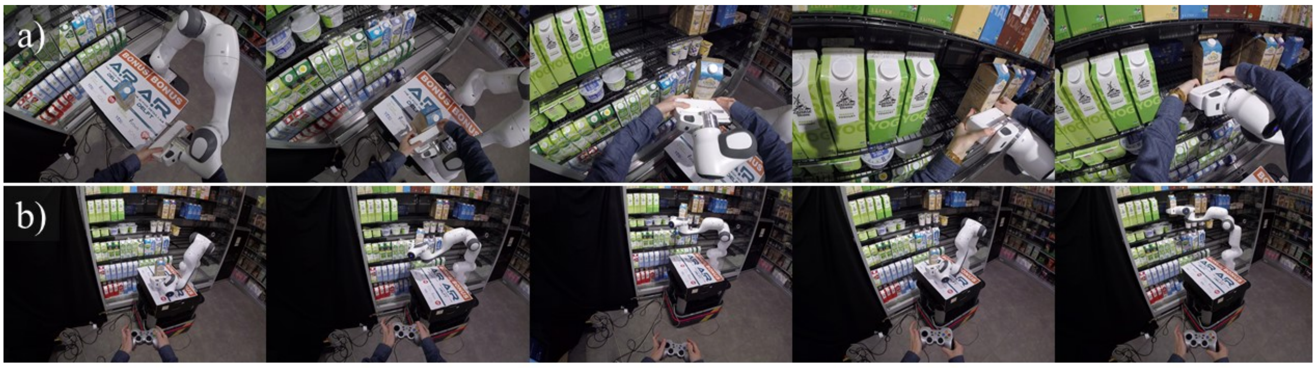


Fig. 2. The learning flow in teaching a robot how to perform item reshelfing.; beginning with a single demonstration in a), followed by multiple rounds of correction in after which the robot is able to autonomously carry out the task as depicted in b).

IV. Validation Experiments

Different experiments were carried out to evaluate the effectiveness, usability as well as robustness of the method. Firstly, the framework's base functionality of taking slow demonstrations and allowing the correction of the dynamics through corrective feedback is tested. A second experiment analyses how well a learned policy can accommodate changes in object properties such as size and weight. Lastly, a user validation study was carried out with non-experts in order to establish the usability of the proposed method.

For our experiments we utilise the 7 DoF Franka-Emika Panda with an impedance controller and a ROS communication network for the online control of the robot with a frequency of 100 Hz. Control of the robot's movement was carried out in real-time, whereas the control of the gripper could not be carried out in real-time due to the internal UDP communication protocol. Furthermore, in order to avoid overloading the database with superfluous points, the recording of the trajectory is carried out at 10 Hz.

A wireless Logitech Gamepad was used for teleoperated corrections due to the number of required inputs. Due to the limited number of reliable, continuous inputs, both the gripper and scaling factor corrections are provided through discrete increments. The attractor corrections are provided through the continuous inputs of the two thumbsticks, with the movement in the x-y-plane regulated by the left thumbstick and the height regulated by the right thumbstick. As an added safety feature, one of the triggers was utilised as a safety button which, when released, ends the execution of the algorithm, halting the robot. Lastly, users can comfortably start the execution from any point along the trajectory as well as bring the robot to the start of the trajectory. Provided a more versatile interface, users could also be given the option to alter the orientation, but this was not the focus of this work and is thus left to future work.

A. *Fast Grasping with MUDS*

For this experiment, a single demonstration was provided wherein the end-effector orientation, gripper width and attractor distance are obtained and used for initialising the respective GP models. The goal of the task is to reduce the execution time by 4 times compared to the time needed to demonstrate the motion with kinesthetic teaching. The experiment was repeated a total of five times.

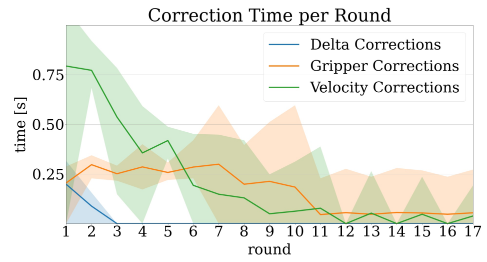


Fig. 3. Range of correction times per round for each aspect depicted by the shaded areas, with the average times depicted by the solid lines.

Within less than three minutes it was possible to fully train the robot to grasp the object in question at the desired performance, four out of five times. Only a fraction of that time was needed for the demonstration and explicit feedback from the human, amounting on average to around 11 s and 6.8 s respectively. This points towards primarily needing fine-tuning corrections from the side of the human, which is further supported by time spent giving corrections for each of the three correctable aspects (see Fig. 3). The time spent correcting the attractor was minimal, as it was only required around the moment when the object is reached. The reason for this is because the human tends to stop at the object during the demonstration in order to avoid knocking it over. In turn, the attractor distance around that point is virtually zero albeit not perfectly zero. Therefore, depending on the demonstration it may happen that this attractor is not pointing in the desired direction. Increasing the magnitude with the scaling factor could thus result in a movement in an unexpected direction. To avoid this, minor corrections to the attractor were provided for ensuring it follows the desired direction. Afterwards only corrections for the gripper and scaling factor are provided. Any time corrections to the scaling factor, i.e. velocity, were provided, corrections to the gripper had to be provided as well. This

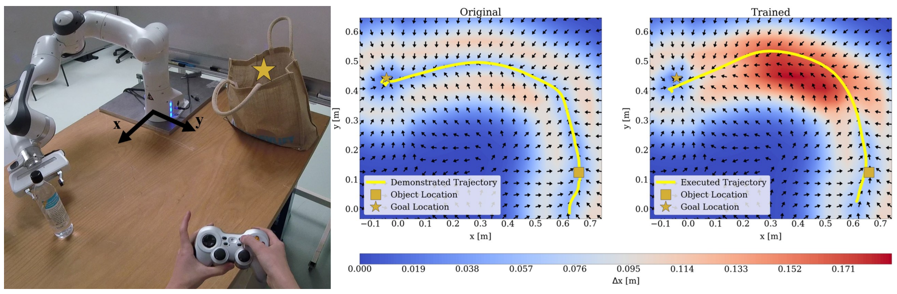


Fig. 4. Use case of robot assistance in grocery refilling. In the attractor vector-field the arrows denotes the direction of the attractor and the color gradient denotes the magnitude of the attractor. The vector field based on original demonstration, with the demonstrated trajectory is compared with the one after training, with the executed trajectory.

TABLE I
METHOD PERFORMANCE

	Demo [s]	**Fdbk** [s]	**Total Time** [s]	**Rounds**	**Success Rate** [%]
Max	11.7	10.324	165.44	17	100
Mean	10.94	6.796	97.47	10.4	82
Min	10.1	4.56	66.61	6	50

was primarily in order to offset the communication delay of the gripper. Once the desired velocity was achieved the final corrections were directed towards fine-tuning the gripper timing. Due to the unreliability of the gripper, despite corrections to the timings the gripper still sometimes closed at the incorrect moment. Nevertheless an average success rate of 82% out of 10 executions could still be achieved. For the complete performance details, please refer to Tab. I. During the experiments, it was established that due to the unreliability of the gripper, it was necessary to push the object for a select period of time, as otherwise the desired performance could not be reached. If the time for pushing the object is too short a very slow motion is needed around the moment of grasping the object, otherwise the grasping success drops to random chance, which could be observed in one of the trials.

One of the main concerns when increasing the velocity along a trajectory is diverging from said trajectory, particularly in curves. While the shape of the trajectory did change slightly, thanks to the uncertainty minimisation, divergence from the trajectory could be avoided even when the attractor magnitude was noticeably increased compared to the original demonstration. This can be observed within the attractor vector fields in Fig. 4.

B. *Generalisation to Object Properties*

Regardless of whether the objects are bigger or smaller, lighter or heavier, rigid or deformable, similar grasping strategies can often be applied to similar objects with minor alterations. Rather than demonstrating and retraining the strategy for every new object, or relying on hard-coded rules in order to adapt to these changes in properties, corrections can be used to adapt the learned policy. To evaluate this, a selection of four different kinds of objects was taken (seen in Fig. 5). First the initial policy was trained on a rigid water-bottle with a weight of 250g. Once a satisfactory policy was achieved, the training object was swapped out for another object. The policy was then executed and corrected if necessary. Corrections were provided until the point that the new object was successfully grasped, after which an evaluation of the performance was performed. Subsequently, a different object was swapped in and the learned policy was reset to the initial policy.

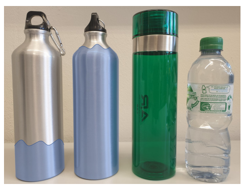


Fig. 5. Left to right: rigid (250g), rigid (900g), flexible (100g), deformable and small (250g)

For each new object, the policy could be successfully corrected. For the same object but with a greater weight, corrections were primarily needed for increasing the velocity around the moment at which the object was grasped. This is due to a larger force needed to move the heavier object at the desired velocity. For the flexible object, the initial policy carried out the grasping successfully in the first execution, hence it was deemed that no corrections were necessary. During the performance evaluation, however, three rollouts resulted in unsuccessful grasps due to the gripper closing too late. Since this issue had not occurred in the first rollout of the policy, additional corrections were not provided. Nevertheless, this could have been improved through additional corrections to the gripper and minor alterations to the velocity. Lastly, for the smallest object it was necessary to reduce the speed of the motion for a successful grasp. Otherwise the object kept being knocked over upon impact due to its smaller support polygon and higher centre of mass. Nevertheless, for all three objects with their different properties it was possible to alter the policy within the time needed for training from a new demonstration, or even within less time if the properties were not too different (see Tab. II).

It is important to note that the strategies for the separate objects are not stored. Retaining this information would require a further form of knowledge representation or policy parametrization, which is outside the scope of the presented work. This evaluation does, however, demonstrate that an existing policy can be corrected in order to generalise to previously unseen objects, which can be beneficial for gathering knowledge more quickly.

TABLE II
OBJECT GENERALISATION PERFORMANCE

	Rigid (250g)	Rigid (900g)	Flexible (100g)	Deformable and small (250g)
Total Time [s]	108.89	31.33	0	124.26
Rounds	7	2	0	10
Success [%]	90	90	70	100

C. User Validation Study

Given that the aim of the proposed method is to enable non-expert users to teach a robot, a preliminary user validation study was carried out. A total of ten participants aged 23 - 28 took part. For this study we made three hypotheses.
H1: Non-expert users will prefer to be able to control the velocity separately from the movement.
H2: Non-expert users will be able to grasp the object and bring it to its designated goal with both methods.
H3: Non-expert users will be able to teach the robot to complete the task within 4 s with both methods.
The final hypothesis was set considering that an expert user is able to achieve a time under 3 s. The same setup as in Fig. 4 was used, with the bag being replaced by a small square tower to provide a clearer goal point for the participants. Participants were given up to half an hour to get familiar with the setup before the actual trials began. There were two trials of ten minutes which were presented in a randomised order to the participants. In one trial, users were required to perform kinesthetic demonstration at a speed that they were comfortable with. After the demonstration, users had the possibility to correct the demonstration with the possibility to scale the attractor distance. The attractor itself was bounded to 4 cm so that the main contribution to the velocity resulting from the scaling factor. Over the course of this section, this trial will be denoted as T1. In the other trial, which will be denoted as T2, users were required to provide a fast demonstration. The attractor for this trial was left unbounded and any corrections that needed to be given for the velocity had to be performed by directly altering the attractor.

In terms of performance all participants were able to successfully grasp the object in T1. Out of these only one participant was unable to reach the 4 s goal. For T2, only one participant was unable to teach the task successfully. The reason behind this was the over-correction of the trajectory which brought the trajectory up against the field generated by the uncertainty minimisation and negated it, placing the robot in a standstill. This error became difficult to correct for two reasons. The first is that since the user was far from the known region and the correlations to the other points were very low, the effect of the corrections were minimal. Therefore, the only effective way to correct the error was to provide corrections before reaching that point. Which brings us to the second point, which is that due to the over-correction of the attractor, the velocity at the point which required large counter-corrections was very high. For someone with little experience with the setup, this can be a challenging situation to correct.

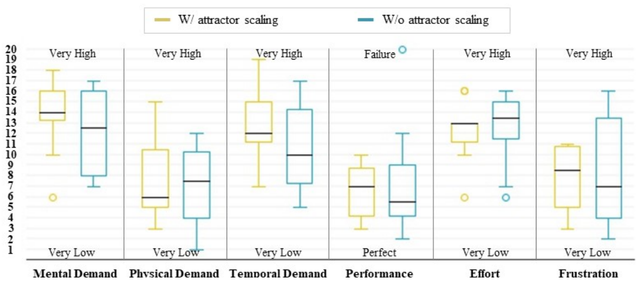


Fig. 6. Results of the NASA TLX questionnaire.

Nevertheless, overall good teaching performance could be observed in both trials. For T1, users were able to teach the task within on average five and a half minutes with 19 correction rounds. The fastest time at which the robot could successfully grasp the object that they were able to teach was 3.4 s with the best time being 2.7 s. For reference, the time needed to demonstrate the behaviour at a fast pace was at best 3.9 s, but generally participants needed more than 5 s to carry out the demonstrations. For T2, user were actually able to achieve better performance on average. In some cases the task could be successfully completed purely with the demonstration. Provided a faster demonstration, the time needed for corrections tended to be lower. Aside from speeding up and timing the gripper, corrections often also had to be provided to slow the robot down in the vicinity of the object if the demonstration had been too fast in that region.

Participants were also asked to fill in a NASA TLX questionnaire to assess the workload of the task for the two versions of the method. The results of the questionnaires were fairly similar for the two methods. Of the more noteworthy, both methods displayed high mental demand where for T1 the mental demand tended to be higher than T2. This can mainly be attributed to the greater number of inputs and the lack of intuitiveness of the input device. At the same time the reported frustration remained in the lower part of the scale, although T1 saw somewhat lower levels of frustration.

Participants were further asked which method they preferred. Between the participants there was no clear preference towards one method or the other. Some preferred to have the unbounded attractor, claiming that it made it easier to provide a nice shape to the trajectory or that it was more intuitive for altering the velocity since it compared more to how joystick inputs were translated to movements of the character. Meanwhile, others found it easier to focus on correcting one aspect at a time, thus preferring to first correct the trajectory before increasing the velocity with the scaling factor. Some of the participants also had the tendency to provide very strong corrections, which in the case of the unbounded attractor resulted in a higher level of frustration and made the overall teaching more difficult for them.

Returning to the initially posed hypotheses, the results of the study showed the following.

TABLE III
PERFORMANCE OF NON-EXPERTS WHO SUCCESSFULLY FINISHED THE TASK

	T1: With Attractor Scaling				T2: Without Attractor Scaling			
	Demo Time [s]	**Training** Time [s]	**Rounds**	**Exec.** Time [s]	**Demo** Time [s]	**Training** Time [s]	**Rounds**	**Exec.** Time [s]
Max	34.1	600	36	4.97	14.9	285	23	4.0
Mean	13.04	323.3	19.4	3.42	8.63	121.22	9.11	2.81
Min	6.4	129	6	2.17	3.9	0	0	2.07

R1: No clear preference could be established within the sample group.
R2: True as out of a total of 20 trials, only a single trial resulted in failure.
R3: True as out of a total of 10 trials for the two methods, only a single trial in each of the methods was unable to reach the desired $4\,\mathrm{s}$ execution time.

V. CONCLUSIONS AND FUTURE WORK

We demonstrate that the dynamics of a user's demonstration can be successfully altered in a non-uniform manner using user corrections. The proposed approach enables the learning of dynamic tasks such as grasping at non-zero velocities without the need of actively detecting the moment of impact. It further allows users to compensate for delays within the system which are not directly known to them but are observable in the system's performance. It was additionally shown that non-experts, irrespective to their prior experience or lack thereof with robots, were able to successfully train the task.

Based on the results of the study, in future investigations the manner in which users can alter the dynamics will remain free for the users to choose according to what they feel most comfortable with. To further improve usability and be able to remedy the effects of over-correcting even when outside the region of certainty, alternative solutions to providing corrections will be investigated. One option would be to determine the most correlated point, and perform the corrections with respect to this point.

Certain aspects remain to be addressed for better generalisation and performance of the proposed framework. A next step would be to enable the correction of the desired orientation. While the current framework would allow for this extension, a better understanding of an intuitive input system is needed. Remaining on the topic of orientation, another aspect to consider is the manner in which the orientation is controlled. Currently, the orientation is provided in accordance to the given position, however, an alternative would be to control according to the current pose and output the desired change in orientation instead of the actual orientation value. This would enable control of the entire pose without the need of a minimum uncertainty mapping.

This formulation could further allow a generalisation of the policy to previously unseen object and goal locations. A further work will focus on expanding the proposed algorithm to accommodate such a generalisation, potentially allowing the framework to be applied in scenarios involving dynamically changing environments.

Lastly, further work will be carried out for allowing an interactive adaptation of the uncertainty minimisation, with the aim of giving users a means of affecting the degree of disturbance rejection.

REFERENCES

[1] G. Franzese, A. Mészáros, L. Peternel, and J. Kober, "Ilosa: Interactive learning of stiffness and attractors," *ArXiv*, vol. abs/2103.03099, 2021.
[2] B. Nemec, A. Ude *et al.*, "Speed adaptation for self-improvement of skills learned from user demonstrations," *Robotica*, vol. 34, no. 12, pp. 2806–2822, 2016.
[3] N. Uchiyama, S. Sano, and K. Ryuman, "Control of a robotic manipulator for catching a falling raw egg to achieve human-robot soft physical interaction," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 777–784.
[4] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, 2016.
[5] I. Havoutis, C. Semini, J. Buchli, and D. G. Caldwell, "Quadrupedal trotting with active compliance," in *2013 IEEE International Conference on Mechatronics (ICM)*. IEEE, 2013, pp. 610–616.
[6] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, 2020.
[7] B. Siciliano and L. Villani, *Robot force control*. Springer Science & Business Media, 2012, vol. 540.
[8] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.
[9] D. Koert, J. Pajarinen, A. Schotschneider, S. Trick, C. Rothkopf, and J. Peters, "Learning intention aware online adaptation of movement primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3719–3726, 2019.
[10] A. Paraschos, C. Daniel, J. Peters, G. Neumann *et al.*, "Probabilistic movement primitives," *Advances in neural information processing systems*, 2013.
[11] B. Nemec, N. Likar, A. Gams, and A. Ude, "Human robot cooperation with compliance adaptation along the motion trajectory," *Autonomous robots*, vol. 42, no. 5, pp. 1023–1035, 2018.
[12] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.
[13] J. Chen and H. Y. Lau, "Learning adaptive reaching skills with nonlinear dynamical systems directly from human demonstrations," in *2016 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*. IEEE, 2016, pp. 232–237.
[14] K. Kronander, M. Khansari, and A. Billard, "Incremental motion learning with locally modulated dynamical systems," *Robotics and Autonomous Systems*, vol. 70, pp. 52–62, 2015.
[15] S. Zimmermann, R. Poranne, and S. Coros, "Go fetch!-dynamic grasps using boston dynamics spot with external robotic arm," 2021.
[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
[17] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "HG-DAgger: Interactive imitation learning with human experts," in *IEEE Int. Conf. Robot. Autom.*, 2019.
[18] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

3 Interactive Learning with GMMs and GPs

Gaussian Mixture Models (GMMs) are a probabilistic model based on the superposition of Gaussian distributions, otherwise referred to as a mixture of Gaussians [1]. Mathematically, this mixture of Gaussians is defined as

$$p(x) = \sum_{k=1}^{K} \pi_x \mathcal{N}(x|\mu_k, \Sigma_k) \tag{3.1}$$

whereby K is the number of Gaussians, also referred to as components, and π_k are the mixing coefficients, which can be interpreted as the weight of each Gaussian. In order to fit the Gaussians to a particular density, the parameters μ_k, Σ_k and π_k have to be determined. This is commonly done through the Expectation-Maximisation of the log-likelihood.

In terms of function approximation, GMMs learn a density $p(x_{in}, x_{out})$ in which x_{in} represents the input data and x_{out} the output data. In order to predict the desired output x_{out} given an input x_{in} one must refer to Gaussian Mixture Regression (GMR). Upon having learned the density $p(x_{in}, x_{out})$, it is possible to determine the conditional density $p(x_{out}|x_{in})$ as

$$p(x_{out}|x_{in}) = \frac{p(x_{in}, x_{out})}{p(x_{in})}. \tag{3.2}$$

When carrying out a regression, however, it is desired to have a single value as the output, this being the mean output value. Using least-squares estimate [4] this can be defined as

$$y_{pred} = \mathbb{E}(x_{out}|x_{in}) \tag{3.3}$$

Using the above formulation, it is possible to determine the output for previously unseen values.

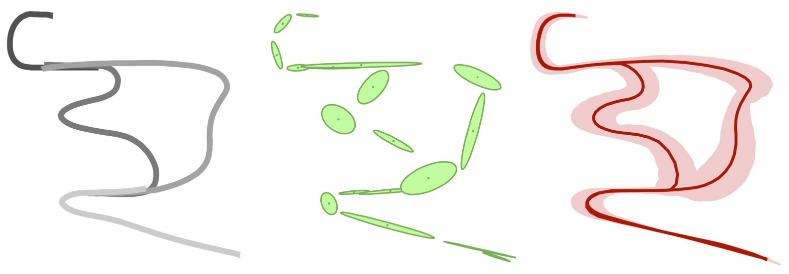


Figure 3.1: Fitting of four trajectories with a GMM with 18 components, followed by their reproduction through regression. [Image taken from [3]]

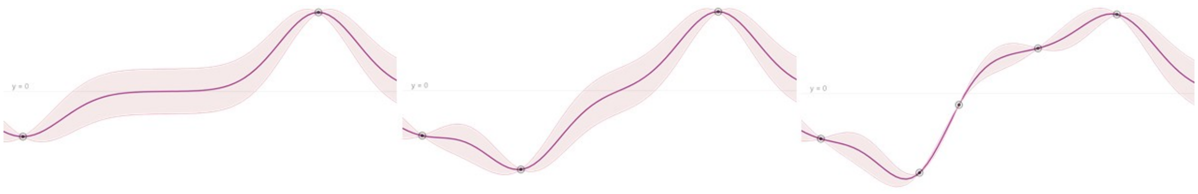


Figure 3.2: Fitting of a GP, provided different amounts of data, with the regression result indicated by the purple line and the uncertainty depicted by the shaded region. Note the lower uncertainty around the training points, as well as when subsequent training points follow the expected trend provided by the prior knowledge encoded in the kernel (image to the right). [Images taken from [8]]

3.1. Comparison to Gaussian Process

Gaussian Processes (GPs) provide the means for making predictions while incorporating prior knowledge. When applying this to the case of regression, one can imagine a region between two demonstrated data-points to have an area of more probable predictions, modelled in the form of a mean and variance [14]. When a previously unseen input is provided, its prediction is then equivalent to the mean and variance in that point of the region. Mathematically, this comes down to

$$\mu = \mathrm{k}_*^\top (\mathrm{K} + \sigma_n^2 \mathrm{I})^{-1} \mathrm{y} \tag{3.4}$$

$$\Sigma = k(x_{new}, x_{new}) - \mathrm{k}_*^\top (\mathrm{K} + \sigma_n^2 \mathrm{I})^{-1} \mathrm{k}_* \tag{3.5}$$

where k_* is the covariance between the new point and the training inputs, K is the covariance matrix of the training inputs, σ_n^2 is the variance of the Gaussian noise at the training points, and y denotes the training output. The predicted mean can then be used as the value of the predicted output, whereas the predicted variance can be used as a measure of uncertainty of the prediction. The output of the covariance is dependent on the chosen kernel function.

Unlike GMMs which learn the joint density function, GPs directly learn the conditional density function $p(x_{out}|x_{in})$. Furthermore, GPs consider the correlation between the training inputs thanks to the use of the kernel function whereas this form of information is not present in GMMs. These correlations can be used for modelling epistemic, i.e. model uncertainty, which are given in the form of the GP's variance. This uncertainty can be utilised as a means to determine the confidence of a prediction. Gaussian Mixture Models on the other hand model aleatoric, i.e. data uncertainty which is dependent on the consistency of the points being fitted by a particular component of the GMM. A further point of difference is that a Gaussian Process is non-parametric, meaning the model is solely dependent on the selected kernel and the training data, whereas a Gaussian Mixture Model is dependent on the number of selected components which can be unintuitive to choose.

3.2. Experimental Comparison

3.2.1. Learning Dynamical Systems: GMM vs GP

In literature, when learning Dynamical Systems in a way that they are not dependent on time, one of the more common methods are Gaussian Mixture Models [11, 15, 13, 5]. Yet, Gaussian Processes, which could provide similar capabilities are not as represented within the literature. For this reason, we investigate the use of both GMMs and GPs for interactively learning a policy.

To this end, the task of grasping an object without altering the velocity was chosen in order to determine the ease of learning the desired attractor distance as well as the reliability of the learned model. Additionally, the behaviour upon leaving the area of the demonstrated trajectory was tested. The control and correction of the gripper was retained from the proposed algorithm in Chap. 2, since this behaviour is seen as separate from the movement dynamics. Similarly, the control of the orientation was retained from the proposed algorithm, as additional demonstrations are not provided to this part of the control. Furthermore, since GMMs do not have a known formulation which allows for the online correction of the policy without the retraining of the GMM, both the policy learned by the GMM and that learned by the GP are modified through local demonstrations wherein one takes control during the execution of the trajectory, inspired by the approach of HG-Dagger [10]. These local demonstrations

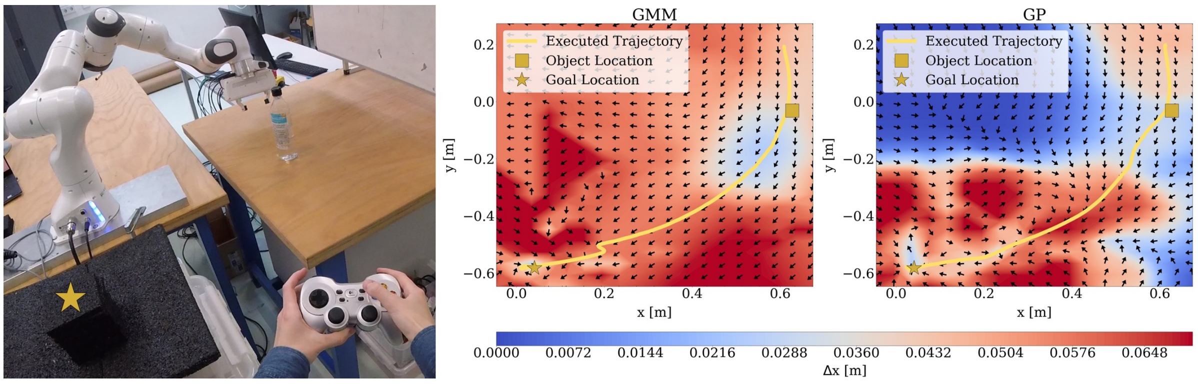


Figure 3.3: Vector-fields generated by GMMs and GPs respectively. The depiction is a projection of the 3D vector-field onto the x-y-plane by taking the z-axis value of closest point for determining the points that are to be evaluated.

Table 3.1: Performance of GMMs vs GPs

	Gaussian Mixture Model			Gaussian Process		
	Training Time [s]	Rounds	Data Points	Training Time [s]	Rounds	Data Points
Max	206.18	13	2472	175.20	9	2579
Mean	132.74	7.6	1443.8	110.00	6.2	1430.8
Min	75.95	5	702	69.86	4	742

are aggregated into the existing dataset and used for the retraining of the models after each episode. The retraining of the models was performed offline, as both in the case of GMMs and in the case of GPs, the training time increased with the size of the dataset, eventually reducing the control rate below the desired 100 Hz.

In terms of ease of learning, overall, GMMs and GPs proved to be comparable in terms of required time and data as seen in Tab. 3.1. The main difference between the two lies in the learned models. Observing the resulting vector-fields in Fig. 3.3 a clear difference can be seen in the length of the attractors. Irrespective of their proximity to the demonstrated samples, GMMs have a tendency to extrapolate the learned behaviour to any point within the workspace. Contrary to this, GPs maintain a region of certainty around demonstrated samples. The further one goes from this region of certainty, the more the predictions will gradually vanish towards the defined mean, which in this case was zero as can be seen in the upper left corner of the GP vector field. This property can aid in ensuring that the robot does not execute any unexpected motions in situations where it has not been taught what to do.

Overall, the performance of the two models could be seen as comparable under the given circumstances. However, for the desired application, GMMs lack certain beneficial aspects which GPs provide. The first aspect is that GPs contain within them a representation of the model uncertainty. This can be exploited for maintaining proximity to the demonstrated behaviour without requiring a separate method for ensuring the same. Secondly, the mathematical formulation of the GPs can be utilised for allowing online correction of the database without the necessity of retraining. This enables faster, more data-efficient and more transparent learning since the provided corrections take effect during the policy execution.

4 Task Parameterization of Dynamical Systems

When grasping an object in everyday life, we generally don't grab objects at the exact same position with respect to ourselves. We are able to adapt if an object is at a different position each time. This chapter provides a short excursion into the topic of policy execution with respect to multiple reference frames with the goal of being able to adapt to different desired object and goal locations. Sec. 4.1 details the strategy and necessary alterations to the previously presented algorithm in Chap. 2. Sec. 4.3 then provides a short validation of the feasibility of the approach and a comparison to Task-Parameterized Gaussian Mixture Models (TP-GMMs) - the state-of-the-art method for trajectory adaptation provided different reference frames. Finally, Sec. 4.4 covers the conclusions which can be drawn from these initial experiments.

4.1. Switching Dynamical Systems

The act of grasping an object and taking it to a desired location can be formulated in two parts. The first part is approaching the object and grasping it, and the second, taking the object to its goal. Even if the motion is fluent, these are two subtasks of the overall task which need to be completed. Thus, our attention tends to first be directed towards the object after which our attention shifts to its desired goal location. Following this reasoning, when the demonstration is initially provided, the trajectory is observed w.r.t. the goal and w.r.t. the object location. These observations are used to train two sets of GP models - one for the goal frame and one for the object frame. When seen in the global frame the initial provided observations are aligned as in Fig. 4.1 a). If either the goal or the object are moved, the known regions of the two models will no longer be aligned as illustrated in Fig. 4.1 b). In order to successfully reach the goal, it is necessary to move towards the known region leading to the goal. The switching of the frames is performed in accordance to the heuristic that once the manipulated object has been grasped, the desire is to bring it to its new location. Furthermore, in order to ensure a smooth switch between reference frames, a short transitioning phase is initiated (Fig. 4.1 c)).

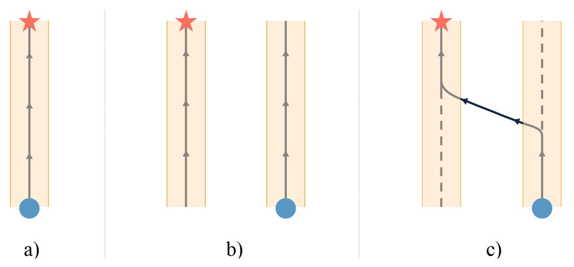


Figure 4.1: a) Initially learned trajectory going from the object (circle) to the goal (star) with the region of certainty depicted by the shade area. b) Learned trajectories with respect to the two reference frames once they've been displaced. c) Transition enabled through the minimisation of the epistemic uncertainty w.r.t. the model leading towards the goal.

Algorithm 1: Transition to different dynamical system

```
21 $\Sigma = \mathrm{GP}_{\Delta x}(x)$
22 if $\Sigma/\Sigma_{\max} > \theta$ and transitioning then
23     $\Delta x_{n,o} = \mathrm{GP}_{\Delta x}(x)$
24     $\gamma_{n,o} = \mathrm{GP}_{\gamma}(x)$
25     $w_{n,o} = \mathrm{GP}^{\delta}_{w}(x)$
26     $[\sin\theta_{n,o}, \cos\theta_{n,o}] = \mathrm{GP}^{\delta}_{\theta}(x)$
27     $[\Delta x, \gamma, w, \sin\theta, \cos\theta] =$ transition($\Delta x_{n,o}, \gamma_{n,o}, w_{n,o}, \sin\theta_{n,o}, \cos\theta_{n,o}$)
28 else
29     $\Delta x = \mathrm{GP}_{\Delta x}(x)$
30     $\gamma = \mathrm{GP}_{\gamma}(x)$
31     $w = \mathrm{GP}^{\delta}_{w}(x)$
32     $[\sin\theta, \cos\theta] = \mathrm{GP}^{\delta}_{\theta}(x)$
33 end
```

Firstly, when the model depicting the behaviour w.r.t. the goal is selected the uncertainty is minimised with respect to that model's variance, which as a result automatically leads the robot towards the trajectory leading to the goal despite previously not having received any demonstrations inbetween the known regions. Secondly, in order to avoid abrupt changes in the predictions while in the region between two trajectories, the uncertainty with respect to the currently selected model is utilised to modulate the predictions. The modulation is merely a weighted average between the predictions of the previously selected model and the predictions of the currently selected model. Mathematically this transition function (l. 28 of Alg. 1) can be written as

$$y = \left(1 - \frac{\Sigma}{\Sigma_{\max}}\right) y_n + \frac{\Sigma}{\Sigma_{\max}} y_o \tag{4.1}$$

where $\Sigma_{\max}$ is the variance of the unconditioned GP with the defined kernel, y_n is the prediction of the currently selected model and y_o is the prediction of the previously selected model. This modulation is used until $\Sigma/\Sigma_{\max}$ falls below a certain threshold, indicating that the robot is once more within the known region.

To enable the transition between different frames of reference, the algorithm presented in the paper in Chap. 2 only requires that ll. 21-23 are expanded into Alg. 1. Only when transitioning between frames are predictions w.r.t. both frames necessary, otherwise the predictions are carried out based on the current frame. Furthermore, when corrections are provided, these corrections are applied to the datasets with respect to the different frames, not only the current one. This ensures that knowledge gained in one frame is transferred to the other frame in accordance to the correlation, reducing unpredictable behaviour such as abrupt changes in the accelerations during frame switching.

4.2. Task Parameterized Gaussian Mixture Model

Gaussian Mixture Models can be expanded to adapt to changes in task parameters by utilising Task Parameterized Gaussian Mixture Models (TP-GMMs) [3]. Examples of task parameters are changes in reference frames or desired via-points.

The main concept is to provide demonstrations for different values of the relevant parameters. Taking the example of different reference frame poses, these demonstrations are observed with respect to both reference frames since the pose of each reference frame is a task parameter. Following the demonstrations, a Gaussian Mixture Model is fitted for each set of observations. These GMMs are then combined through the product of the corresponding components and taking the resulting Gaussian distributions in order to yield the final model.

The product of two Gaussian distributions is given by

$$c\mathcal{N}(\mu^P, \Sigma^P) = N(\mu^{(1)}, \Sigma^{(1)}) \cdot N(\mu^{(2)}, \Sigma^{(2)}) \tag{4.2}$$

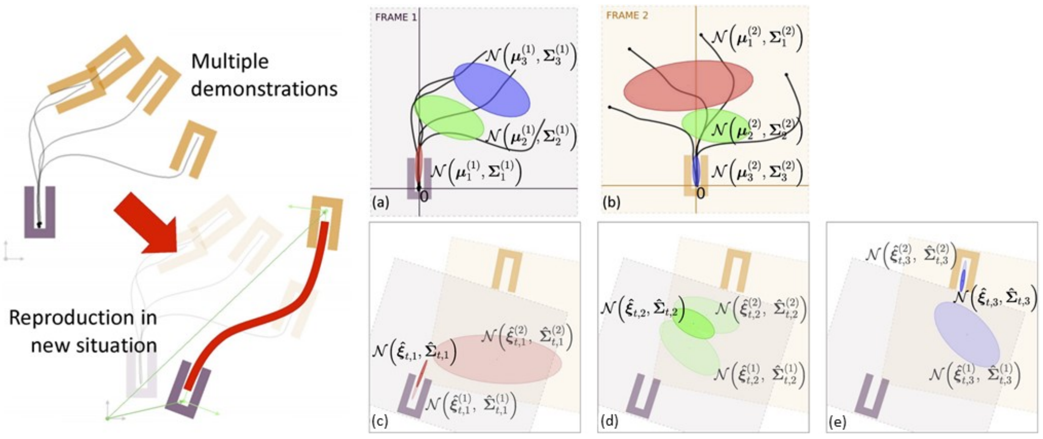


Figure 4.2: (a) - the multiple provided demonstrations as seen from Frame 1 with the fitted GMM components, (b) - the multiple provided demonstrations as seen from Frame 2 with the fitted GMM components, (c)-(e) - linear transformation of the components base on the new task parameters and the results of the products of the component pairs

with

$$\begin{aligned} c &= \mathcal{N}(\mu^{(1)}|\mu^{(2)}, \Sigma^{(1)} + \Sigma^{(2)}) \\ \Sigma^P &= (\Sigma^{(1)^{-1}} + \Sigma^{(2)^{-1}})^{-1} \\ \mu^P &= \Sigma^P(\Sigma^{(1)^{-1}}\mu^{(1)} + \Sigma^{(2)^{-1}}\mu^{(2)}) \end{aligned}$$

where only $\mathcal{N}(\mu^P, \Sigma^P)$ is considered for the final model. Prior to carrying out this product, one must transform the GMMs that were learned for each task parameter to a general representation in which also newly chosen values for said task parameters are discernible. In the case of reference frames, one learns the GMMs with respect to each frame, then prior to the computation of the product of the Gaussians, the learned Gaussian components are linearly transformed with respect to the newly chosen poses of the frames. For a clearer visualisation please refer to Fig. 4.2.

4.3. Experimental Validation

Being able to grab an object should further be generalisable to different object locations. Whether an object is a bit further to the left or a bit closer than the original demonstration should not have a large effect on the overall grasping strategy. Similarly, if the desired goal at which the object should be placed also changes, the robot should be able to adapt to this. To this end, pairs of GP models for each aspect which has to be controlled (attractor distance, gripper width, and orientation) have to be learned. One model out of the pair provides predictions w.r.t. to the object's initial location, whereas the other model provides predictions w.r.t. to the goal location. For these experiments, the object

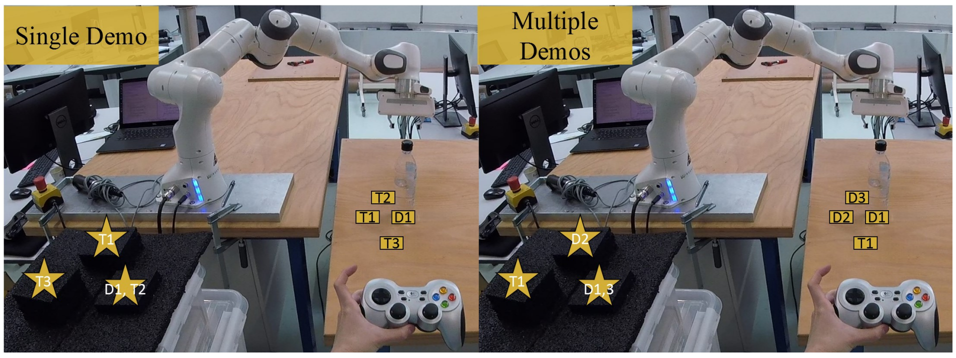
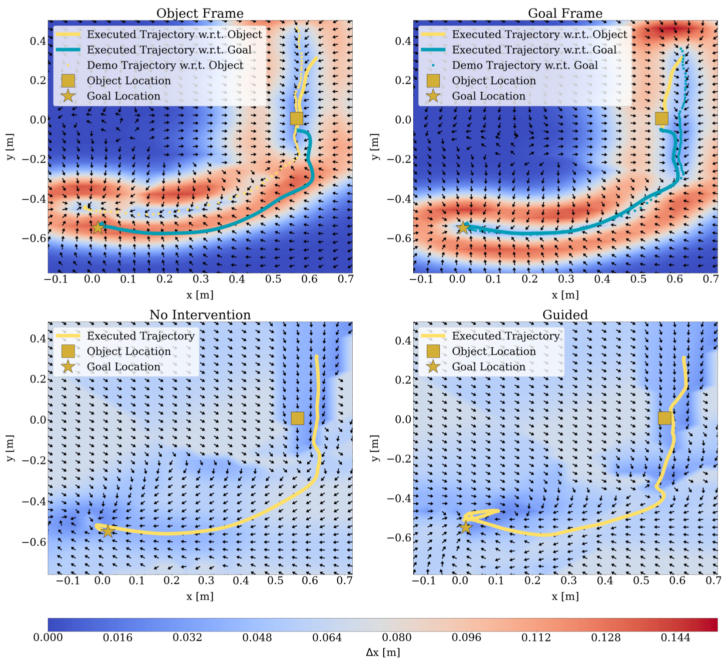


Figure 4.3: The experimental setup. Object and corresponding goal locations used for the demonstrations are denoted by D, whereas the object and goal locations used for the evaluation are denoted by T. While object locations are only varied in the plane, goal locations are varied within the 3D space.



(a) Vector field and executed trajectory of the proposed framework utilising GPs.

(b) Vector field and executed trajectory of the TP-GMMs. Due to lack of a stabilisation field for TP-GMMs, a check of the learned performance was carried out both without any form of intervention and with the user physically guiding the robot towards the known region.

Figure 4.4: Comparison of vector fields and executed trajectories of GPs (a) and TP-GMMs (b) provided a single demonstration and corrections for the execution of this demonstration.

frame is fixed to the initial location of the object given that no system was incorporated for tracking it in real time.

A common strategy to adapt to changes in task parameters, such as for example the locations of reference frames, is enabled through the use of TP-GMMs. Thus a comparison between this state-of-the-art and the newly proposed method is carried out. Since the focus is on the trajectory generalisation, velocity modulation is left out for this experiment. Additionally, for the same reason, only the model predicting the attractor distance is replaced by TP-GMMs. In all of the experiments, the gripper width and end-effector orientation continue to be controlled by the GP models.

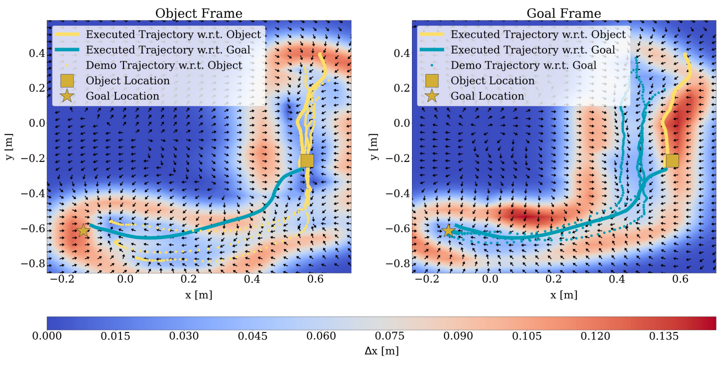
The TP-GMMs were trained using the same inputs and outputs of the GP model, i.e. the robot position as input and the desired attractor distance as output. The orientation of the frames was kept constant during the experiments and was thus not taken into consideration. The number of components for the GMMs needed for trainging the model were tuned based on the observed performance, with 8 components providing the best results. Two sets of GMM models were trained, one w.r.t. the object frame and one w.r.t. the goal frame, much like the GP models. The matching of the GMM components of the two models for the product of Gaussians was performed by taking the closest component.

Two versions of the experiment are carried out. In the first version, a single demonstration is provided, whereas in the second version, three demonstrations are provided with the object and goal in different locations. The reason for this variation is that GMMs tend to perform better with more demonstrations to cover the possible workspace, whereas the proposed method tends to perform better with a single demonstration since the region of certainty is narrower resulting in a more prominent effect of the uncertainty minimisation. After the demonstrations, interactive corrections were provided. For the proposed method it was through the proposed online corrections, whereas the TP-GMMs were corrected through local demonstrations which were aggregated and used for retraining the model after each rollout. The time limit for providing corrections for each model was set to 10 minutes.

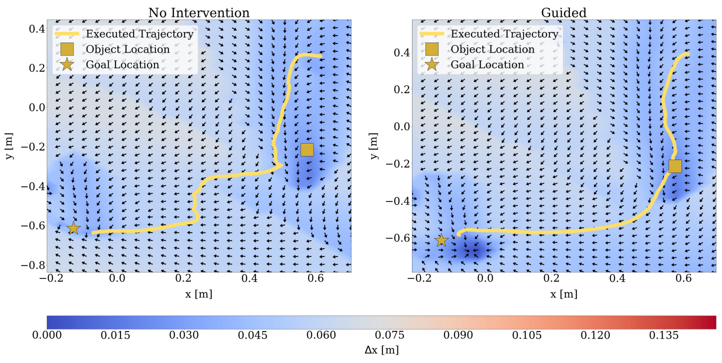
4.3.1. Single Demonstration

In the experiment with a single demonstration for GPs it could be observed that the behaviour followed a fairly precise motion, mainly thanks to the narrow certainty region which can be observed in Fig. 4.4 a). At the moment the object was grasped, and the frame switched, the uncertainty minimisation was able to successfully bring the robot into executing the desired motion for reaching the goal.

In the case of TP-GMMs, although good performance could be achieved along the known trajectory,



(a) Vector field and executed trajectory of the proposed framework utilising GPs.



(b) Vector field and executed trajectory of the TP-GMMs. The same check for the performance of the TP-GMMs was performed as in the case of the single demonstration experiment.

Figure 4.5: Comparison of vector fields and executed trajectories of GPs (a) and TP-GMMs (b) provided a multiple demonstrations and corrections for the execution of these demonstrations.

the moment another object and goal position was requested the model either lacked the necessary precision or completely diverged from the demonstrated trajectory. Due to being in an unknown region and the extrapolation performed by the underlying GMMs the robot begins to stray from the demonstration. This is largely due to the fact that TP-GMMs have no manner of returning to the known region on their own. To verify that this is indeed the case, the TP-GMMs were evaluated once without any form of intervention and a second time through a physical intervention in which the robot was pulled towards the region of the demonstrations. Observing the vector fields in Fig. 4.4 b) it could be noticed that this is indeed the case. However, in order to ensure proximity to the known region, a separate method would have to be introduced, whereas in the case of the proposed framework only the existing information of the epistemic uncertainty is required.

4.3.2. Multiple Demonstrations

Meanwhile, in the experiment with multiple demonstrations GPs displayed a slightly less precise behaviour on the approach towards the object. This is due to the three separate demonstrations w.r.t. the object covering a broad area in the workspace in turn translating to a broad certainty region as the one in Fig. 4.5 a). The result of this is that the uncertainty minimisation begins acting on the robot later than in the case of the single demonstration. Despite this, the policy was able to successfully generalise to an unknown object and goal location. The training time was naturally longer than in the case of a single demonstration as each of the three demonstrations had to be evaluated and slightly corrected for ensuring the succesful completion of the task. TP-GMMs displayed similar issues as those observed in the case of a single demonstration. Nevertheless, thanks to the multiple demonstrations, a larger portion of the workspace was covered thus reducing the tendency stray from the demonstrated trajectories. It was nevertheless not precise enough to accomplish the task on its own.

In conclusion, the proposed method in the given setting is capable of generalising to new frames using only the provided information without requiring additional information for a separate algorithm to ensure proximity to the known region. A further advantage is that the proposed method is already capable of generalisation provided a single demonstration.

4.3.3. Limitations

There are currently still some limitations in terms of the range of generalisation. Firstly, the new unknown locations have to be fairly close to the demonstrated points. If the object were to for example be moved to the other end of the workspace from the demonstrations while keeping the same initial starting position of the robot the robot would find itself too far from the known region. In such a case, the uncertainty minimisation would have no effect as it is within the plateau of maximum uncertainty resulting in a gradient of zero. A good point however is that at the same time, due to the vanishing effect of the GPs the robot would remain at a standstill given the attractor distance would also be zero thus preventing unexpected motions. This limit of generalisation could potentially be remedied in two ways; either having a separate mechanism which brings the robot to the beginning of the trajectory or enabling interactive corrections which add new datapoints as in ILoSA. With the second option the robot would be lead through the unexplored region of the workspace, in turn gathering more information for a more feasible approach while simultaneously remaining more data-efficient than simple data aggregation.

The second limitation is that the current method does not consider frame orientation. In the case that an object has to be grasped from a particular side, the policy would not be able to adapt to it. This is something which TP-GMMs, however, already consider and are able to generalise to. One potential approach to incorporation generalisation to orientation would be to not only control with respect to the Cartesian position but with respect to the complete Cartesian pose and rather than providing the desired orientation, providing the desired change of orientation in the current state. Nevertheless, this remains to be explored in future works.

It should also be noted that TP-GMMs have the property of being able to determine w.r.t. which frame the robot should be moving based on the demonstration data rather than explicitly selecting the current frame. In our proposed method, the decision regarding which frame to take as reference is handled by the heuristic of having grasped the manipulated object. A higher level frame selector would have to be taken, in order to determine which frame to choose in a general setting. This selector could either be learned implicitly based on information from provided demonstrations [12, 9] or explicitly learned through user demonstrations and active corrections as in [7]. This also remains to be explored further.

4.4. Conclusions and Future Work

With minimal alterations to the base framework, it was possible to enable the generalisation to different object and goal positions. When reference frames are switched, the minimisation of the epistemic uncertainty automatically enables the transition to the newly desired behaviour. At the same time, utilising the information of the uncertainty, the transition between the predictions could be carried out smoothly instead of having a discrete switch between prediction models.

Certain important aspects remain to be addressed for the better generalisation of the proposed framework. Currently, the orientation is provided in accordance to the given position, however, an alternative would be to control the change in orientation provided the current pose. This could potentially also enable a way to control the robot's orientation w.r.t. to a given reference frame. Further work also remains to be done on the manner of selecting the reference frames. Different possibilities exist, from learning the desired frame of reference from user corrections, to utilising TP-GMMs as a classifier for the desired reference frame or using another high-level selection algorithm. Due to the first promising results, work will be continued in these directions and will potentially be expanded to use-cases dealing with dynamic environments.

A Appendix - Learning Variable Orientation Control

When deciding on the manner in which to control the orientation with impedance control for a redundant robot, there are two possible approaches. The first option would be to only control the Cartesian position based on the impedance equation, while using the null-space to control the orientation based on the joint configurations. The second option would be to provide the controller with the desired orientation directly. Since the orientation is to be fitted with a Gaussian Process, certain considerations have to be made.

A.1. Orientation Control with Joint Configurations

When dealing with redundant robots, one can control the redundant degrees of freedom through the use of null-space control. Taking the example of Cartesian impedance control, incorporating null-space control effectively allows one to minimise the present error by generating the torques

$$\tau = J^{\top} f = J^{\top}(-K_s \cdot \epsilon - D(J \cdot \dot{q})), \tag{A.1}$$

where K_s is the Cartesian stiffness, D is the corresponding critical damping, ϵ is the pose error, $\dot{q}$ is the joint velocity and J the Jacobian of the end-effector pose, while at the same time allowing for a secondary control goal. This secondary control goal can be carried out while ensuring to not have effect on the dynamics of the end-effector, i.e. $f = J^{\top\dagger} * \tau_n = 0$. This equation would be satisfied for τ_n equal to zero or with the help of the projection onto the null-space [2] according to

$$\tau_n = (I - J^{\top} \cdot J^{\top\dagger})u_n,$$

where I is identity, $J^{\top\dagger}$ is the pseudo-inverse of $J^{\top}$ and u_n is the control input in the null-space. For the null-space joint impedance controller this translates to

$$\tau_n = (I - J^{\top} \cdot J^{\top\dagger})(K_n \cdot (q_{goal} - q) - 2\sqrt{K_n}\dot{q}). \tag{A.2}$$

What Eq. A.2 does is generate torques which will only move the joints that will not affect the dynamics of the end effector regulated by Eq. A.1.

In the present setup, the impedance controller allows the control of the entire pose with one redundant degree of freedom. The system Jacobian is a 6-by-7 matrix, with the first three rows corresponding to the position representation and the last three rows corresponding to the orientation representation. In order to control the orientation as a function of the joint configuration we can use the null-space of the Jacobian which only considers the position, i.e. a 3-by-7 matrix.

This approach has two benefits. The first is that by controlling the configuration we not only control the orientation but we also fully control all the degrees of freedom of the robot and as a result ensure cyclicity [6], which is important for a reliable actuation of the robot for periodic tasks over a long span of time. The second is that the joint angles are continuous when recorded from a kinesthetic

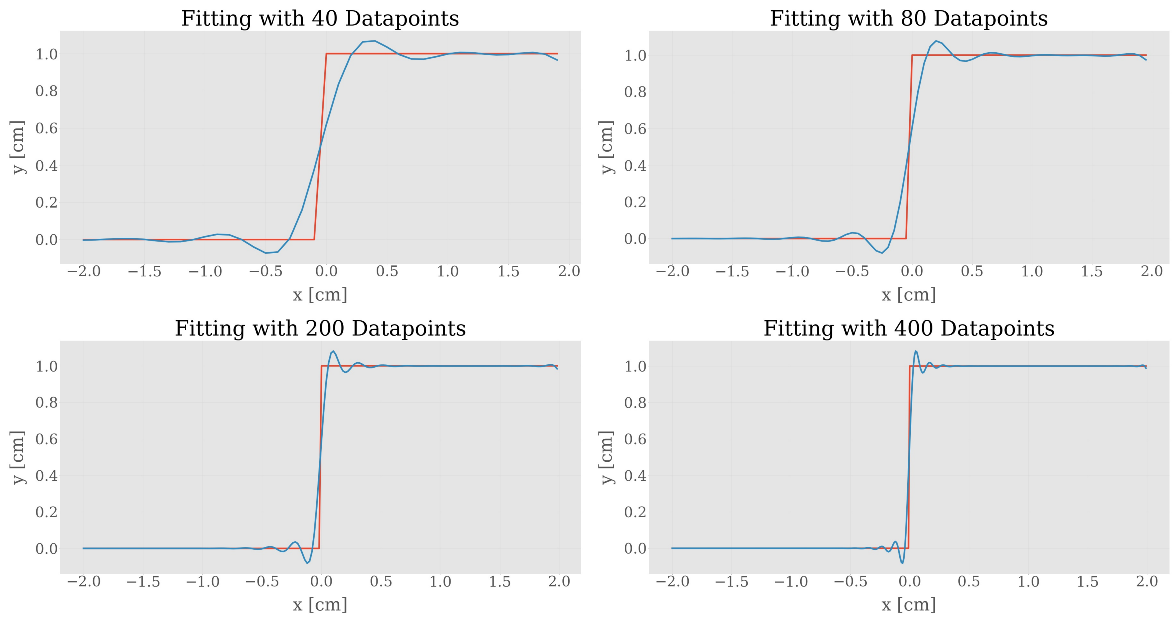


Figure A.1: Results of fitting a step function with a Gaussian Process using different amounts of datapoints.

demonstration, which is beneficial for the modelling with a Gaussian Process - more on this in the upcoming section.

While this initially achieved good results, it was later discarded on the grounds of limited generalisability. In the case that the trajectory had to be executed in a different part of the workspace, in order to maintain the same orientation and not generate conflicting torques with the ones controlling the position, the joint policy had to be translated to the new desired pose. This involves carrying out forward kinematics on the joint dataset to obtain the desired orientation. This orientation would then be combined with the desired global positions in order to perform inverse kinematics to obtain the new joint configurations. While this could be performed before the trajectory execution in order to save on computation time during the policy execution, it would have to be performed any time the object or goal position would be changed. This would, however, make policy execution in the case of a moving object or goal impossible at high frequencies.

A.2. Rotation Representations

In robotics there are different manners of representing rotations in 3D Cartesian space, however, some representations such as quaternions and Euler angles displayed a discontinuous jump in their values upon very slight changes of the robot configuration such as a flipping of the signs in the case of the quaternions or switches from π to $-\pi$ in the case of Euler angles. Due to the smoothness of the GP kernels, such discontinuities cannot be fitted well and often result in inaccurate predictions around the discontinuity. A visualisation of this can be seen in Fig. A.1. Even with 80 datapoints across 4cm the smoothness of the Gaussian Process results in a poor fitting of the step function. This effect can be reduced by increasing the number of datapoints, however, this will eventually lead to an increasingly large database as the demonstrated trajectories ranged to more than a meter in length. Therefore, such an approach would not be practical due to a high computational load even though the training of the GPs is only carried out once following the kinesthetic demonstrations. For this reason, a continuous representation is desired.

Out of the common representations, rotation matrices provide precisely this. While the final rotation matrix which is used as part of a transformation matrix has nine unique values, the number of inputs needed to represent the orientation can be reduced to six. These six values are the $\sin$ and $\cos$ values of the Euler angles from the rotation matrices used to obtain the final rotation matrix according to

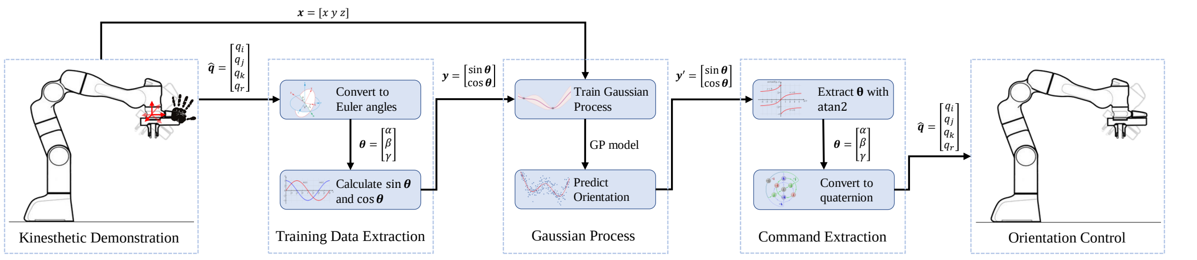

$\boldsymbol{x} = [x\ y\ z]$
$\hat{\boldsymbol{q}} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix}$
Convert to Euler angles
$\boldsymbol{\theta} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$
Calculate sin $\boldsymbol{\theta}$ and cos $\boldsymbol{\theta}$
$\boldsymbol{y} = \begin{bmatrix} \sin\boldsymbol{\theta} \\ \cos\boldsymbol{\theta} \end{bmatrix}$
Train Gaussian Process
GP model
Predict Orientation
$\boldsymbol{y}' = \begin{bmatrix} \sin\boldsymbol{\theta} \\ \cos\boldsymbol{\theta} \end{bmatrix}$
Extract $\boldsymbol{\theta}$ with atan2
$\boldsymbol{\theta} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$
Convert to quaternion
$\hat{\boldsymbol{q}} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix}$
Kinesthetic Demonstration
Training Data Extraction
Gaussian Process
Command Extraction
Orientation Control

Figure A.2: An dataflow illustration of the orientation data, beginning with the data recording and conversions for training the prediction model, followed by extracting the predictions and converting back to the desired control command.

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}. \tag{A.3}$$

During a continuous demonstration of the orientation, the resulting $\sin$ and $\cos$ values will remain continuous. With the requirement of a continuous representation fulfilled, while at the same time remaining fairly compact the three [$\sin$, $\cos$] pairs were chosen for learning the final orientation model. This representation does bring with it a few additional steps, which will be outlined briefly in the next section.

A.3. Dataflow

The Cartesian impedance controller provided by Franka takes quaternions as inputs for the desired orientation and also allows the orientation to be read in the form of quaternions. Thus, to obtain the desired representations, the following steps had to be taken.

Firstly, during the demonstration, the orientation at each position is recorded in quaternion form. These quaternions are then converted into Euler angles of which the $\sin$ and $\cos$ are calculated.

The six resulting values are taken as the outputs of the orientational GP model, with the inputs being the Cartesian positions. When the GP is used for predicting $[\sin\alpha, \sin\beta, \sin\gamma, \cos\alpha, \cos\beta, \cos\gamma]$, the Euler angles α, β, and γ have to be extracted. This can be done by calculating the $\mathrm{arctan2}$ as

$$\theta = \mathrm{arctan2}\left(\frac{\sin\theta}{\cos\theta}\right). \tag{A.4}$$

Finally, the Euler angles are once more converted into quaternions and provided as part of the control input to the low level impedance controller.

B
Appendix - Constrained optimisation of GP Kernel Length Scales

One of the advantages of Gaussian Processes over Gaussian Mixture Models is that GPs are non-parametric. Nonetheless, the kernel chosen for a particular GP model can change the model's overall performance. Fortunately, the selection of the kernel parameters is fairly straightforward in the given setting. For this work an Automatic Relevance Determination (ARD) squared-exponential kernel [14] with added noise was chosen. The added noise ensures that the model does not overfit to the presented data.

An ARD squared-exponential kernel of dimension d takes the form of

$$k(x, x') = \sigma_f^2 \exp\left[-\frac{1}{2}\sum_{m=1}^{d}\frac{(x_m - x'_m)^2}{\sigma_m^2}\right] \tag{B.1}$$

where σ_m represents the characteristic length scale for each dimension m, σ_f is the signal standard deviation, x is the known point and x' is the point to be evaluated. In our particular case, the total number of dimensions is $d = 3$, given we are using the 3-D Cartesian position.

There are two cases in which it was necessary to bound the length scales. The first is in the case that the trajectory remained mostly constant along a particular axis. One can argue that in the case this happens, when for example moving along more or less the same height, the correlation of the inputs is independent of the height. This is precisely what the ARD kernel displays. What then happens during the parameter optimisation, is that the length scales of such dimensions become very large. While this achieves the goal of an ARD kernel, which is to essentially remove the effect of irrelevant variables, this generates a very broad region of certainty along this axis. What this means, is that the uncertainty minimisation intended for remaining close to the trajectory becomes virtually ineffective along this axis. To remedy this, an upper bound had to be introduced to the kernel modelling the translational dynamics. Since the length scales in the more variable axes tended to be between ten and fourteen centimetres, the upper bound was set to 15cm.

The second case is when it is undesirable that a correction carried out in one point affects the outputs of surrounding datapoints. This was the case for the gripper. If the length scales were left to be as large as those of the model for the translational dynamics, the effect of the corrections would spread to earlier points of the trajectory. As a result it could easily happen that the gripper closes sooner than what the user was intending due to the gripper not being controlled in real-time. To this end, the length scale of the gripper was set to an upper bound of 2cm, practically limiting the user's corrections to the current datapoint used for the prediction.

C

Appendix - User Interface

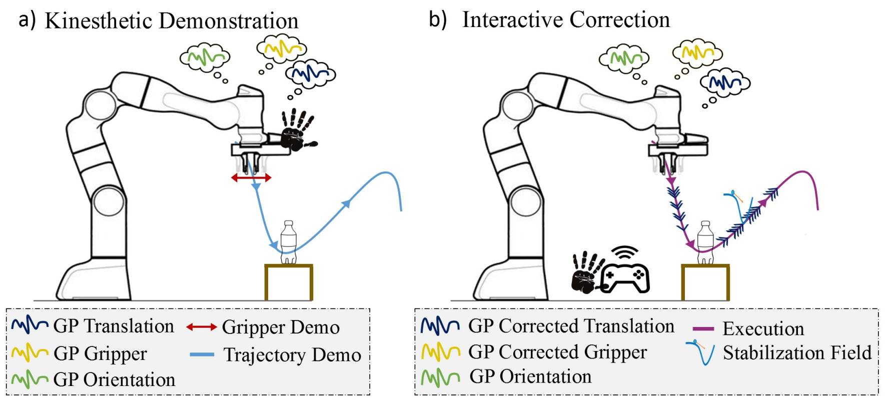


Figure C.1: Overview of the teaching approach.

The user interface consists of two parts, the kinesthetic teaching and the teleoperated corrections, depicted in Fig. C.1. For the kinesthetic demonstration, the stiffness at the robot's end-effector is reduced to zero which allows the user to freely move the robot. The gravity compensation implemented within the impedance controller ensures that the robot does not fall under its own weight. During the demonstration, the desired motion, orientation and gripper behaviour are provided. Subsequently, the policy is executed and the user is allowed to provided corrections to the dynamics of the translational motion and gripper behaviour. To ensure that a user was able to comfortably provide the necessary corrections, an adequate teleoperation device had to be selected.

C.1. Teleoperation Device

For the purpose of testing the usability of the framework with non-expert users an input device was needed which fulfilled the following requirements:

- readily available\not too expensive
- simple setup process
- adequate number of inputs for providing all of the necessary forms of correction
- portable in order to allow users to move around if they need a better angle to see the robot

For this reason, the F710 Logitech Gamepad was selected despite not necessarily being the most intuitive out of the existing options. The commands enabled for the users were organised as depicted in Fig. C.2.

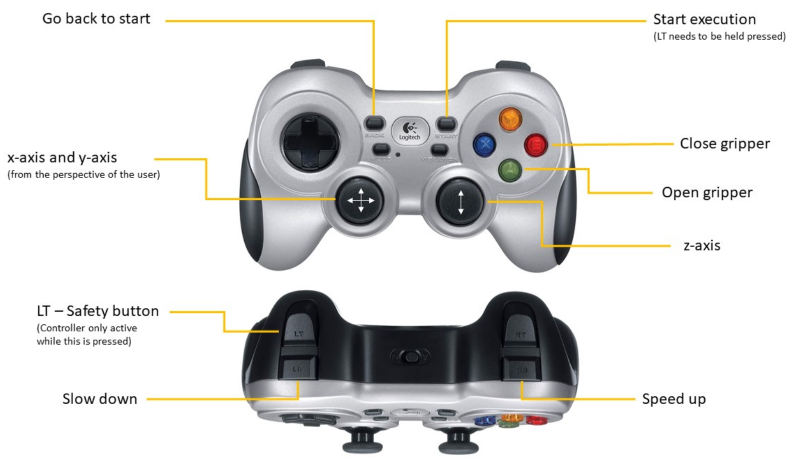


Figure C.2: Controller commands

In order to maintain the flow of the teaching without needing intervention by the researcher conducting the study (so as to e.g. bring the robot back to the start of the trajectory), users were given full control over the system. With the Back button, users could bring the robot to the start of the trajectory from any point in the workspace, provided the execution of the policy is terminated. With the Start button, the execution of the policy could be started from any point within the workspace as long as the safety button, i.e. the Left Trigger, was pressed. The policy would only be executed as long as the safety button was held pressed. The safety button was in no way a replacement for the actual emergency button of the Panda. Rather, the safety button was primarily there to enable users to terminate the execution the moment the robot happened to do something unexpected. During the experiments, the researcher present had the emergency button within reach, although it did not have to be used over the course of the study.

In terms of actual corrections, users had a total of seven inputs - three for the attractor distance Δx, two for the scaling factor γ and two for the gripper width w. When assigning the commands to the buttons on the controller, considerations were made towards the typical functions of controller inputs for video games. The reasoning behind this was that people who play video games would struggle more when provided with controls in locations they're not familiar with, than people who don't play video games.

The inputs for the attractor distance could be provided as continuous values, although they had to be split across the two thumbsticks since the inputs of each thumbstick are two-dimensional. Since in most video games the left thumbstick is used for moving around, this one was assigned for attractor corrections in the x-y-plane and the corrections along the z-axis were assigned to the right thumbstick.

For speeding up and slowing down, one of more common inputs are the Left and Right Triggers. However, there appeared to be an issue of the initialisation of the trigger values, such that if the controller happened to be reinitialised the value being read was 0 whereas after the trigger was pressed and let go this value would be -1. While this value could be offset and the users requested to press the triggers prior to each round of correction, the triggers were discarded in order to avoid any potentially unexpected behaviour and to avoid adding further mental load on the users. Instead, this functionality was shifted to the two buttons above the triggers. Only discrete inputs could be provided in order to increase or decrease the velocity. Nonetheless, if the user kept the button pressed, the input would be continually provided as long as it remained pressed.

Out of all the buttons on the controller, the A, B, X, and Y buttons are the most variable across games and platforms and are used for carrying out different kinds of actions. Thus, the gripper corrections were simply set to the A and B buttons. These buttons also only allow discrete inputs, but function in the same manner as the ones intended for increasing and decreasing the velocity. Therefore, for each registered input, the width of the gripper was incrementally altered.

Bibliography

[1] Christopher Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[2] Samuel R Buss. “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods”. In: IEEE Journal of Robotics and Automation 17.1-19 (2004), p. 16.

[3] Sylvain Calinon. “A tutorial on task-parameterized movement learning and retrieval”. In: Intelligent service robotics 9.1 (2016), pp. 1–29.

[4] Sylvian Calinon. Machine Learning for Engineers - Nonlinear Regression. Nov. 2015.

[5] J. Chen and H. Y. K. Lau. “Learning adaptive reaching skills with nonlinear dynamical systems directly from human demonstrations”. In: 2016 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO). 2016, pp. 232–237.

[6] Stefano Chiaverini, Giuseppe Oriolo, and Ian D. Walker. “Kinematically Redundant Manipulators”. In: Springer Handbook of Robotics. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 245–268. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_12. URL: https://doi.org/10.1007/978-3-540-30301-5_12.

[7] Giovanni Franzese, CE Celemin, and Jens Kober. “Learning interactively to resolve ambiguity in reference frame selection”. In: Conference on Robot Learning (CoRL). 2020.

[8] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. “A Visual Exploration of Gaussian Processes”. In: Distill (2019). https://distill.pub/2019/visual-exploration-gaussian-processes. DOI: 10.23915/distill.00017.

[9] Jose Hoyos et al. “Incremental learning of skills in a task-parameterized gaussian mixture model”. In: Journal of Intelligent & Robotic Systems 82.1 (2016), pp. 81–99.

[10] Michael Kelly et al. “HG-DAgger: Interactive imitation learning with human experts”. In: IEEE Int. Conf. Robot. Autom. 2019.

[11] S. Kim, A. Shukla, and A. Billard. “Catching Objects in Flight”. In: IEEE Transactions on Robotics 30.5 (2014), pp. 1049–1065.

[12] Scott Niekum et al. “Learning and generalization of complex tasks from unstructured demonstrations”. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012, pp. 5239–5246. DOI: 10.1109/IROS.2012.6386006.

[13] A.S. Phung et al. “Learning to Catch Moving Objects with Reduced Impulse Exchange”. In: IFAC Proceedings Volumes 47.3 (2014), pp. 3036–3041.

[14] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning. The MIT Press, 2006. ISBN: 026218253X.

[15] S. S. M. Salehian, M. Khoramshahi, and A. Billard. “A Dynamical System Approach for Softly Catching a Flying Object: Theory and Experiment”. In: IEEE Transactions on Robotics 32.2 (2016), pp. 462–471.