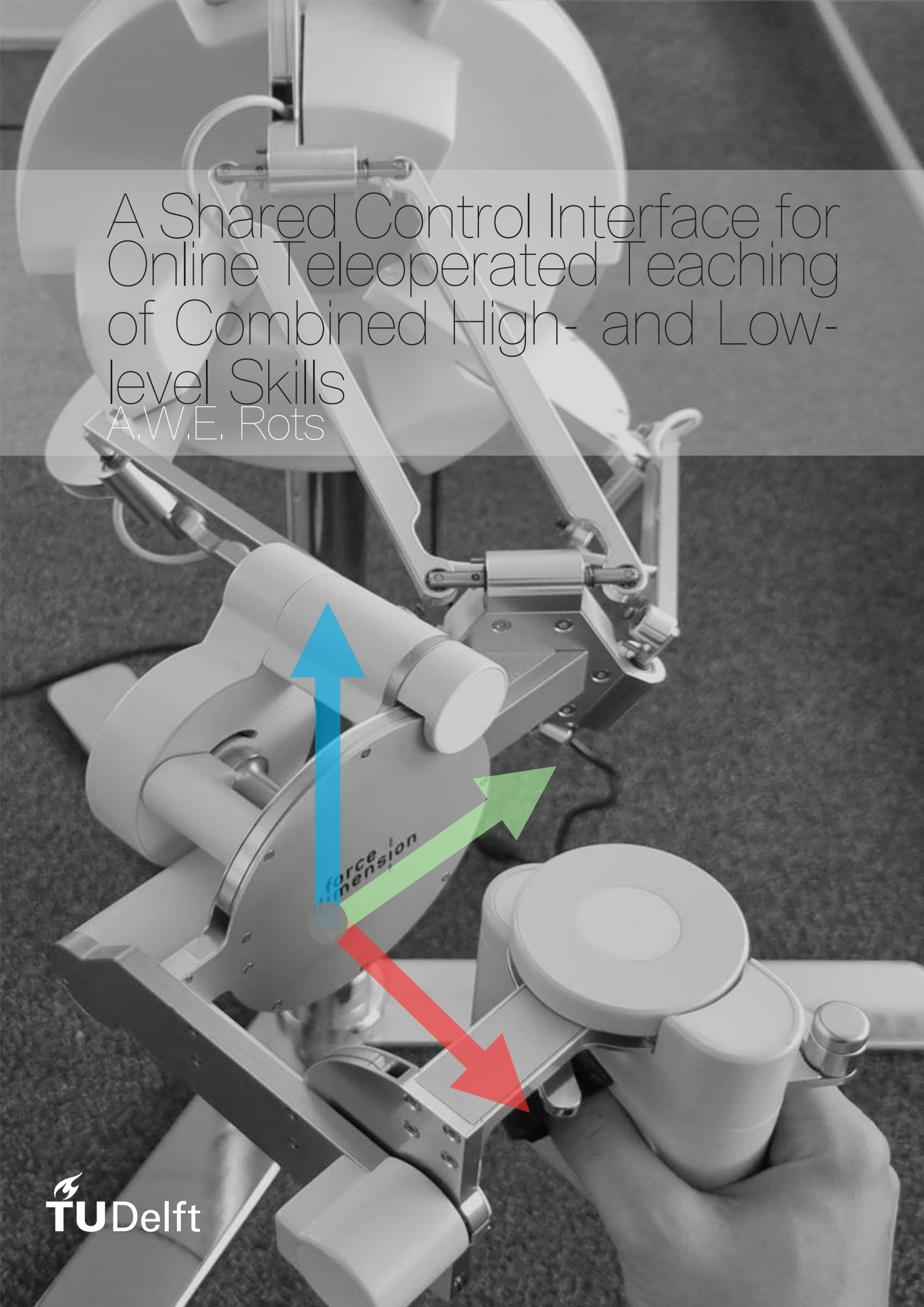
A Shared Control Interface for Online Teleoperated Teaching of Combined High- and Low-level Skills

A.W.E. Rots



TU Delft

A Shared Control Interface for Online Teleoperated Teaching of Combined High- and Low-level Skills

by

A.W.E. Rots

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday October 11th, 2023 at 1:00 PM.

Student number:	4589726	
Project duration:	February 13, 2023 – October 11, 2023	
Thesis committee:	Dr. ir. L. Peternel,	TU Delft, supervisor, chair
	Prof. dr. ir. D.A. Abbink,	TU Delft, member
	Dr. ir. W. Mugge,	TU Delft, member

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis marks the conclusion of my Master's in Robotics at Delft University of Technology. Over the past eight months, I developed a novel shared control interface for teleoperated teaching of combined high- and low-level skills.

I would like to thank my supervisor, Luka Peternel, for his invaluable guidance and advice during my literature study and thesis. He was always ready to help. His critical thinking, encouragement and support ensured that I left our weekly meetings feeling challenged, full of new ideas and positive energy. Of course, I want to thank Professor Abbink and Winfred Mugge for making time to review my thesis as committee members. During the project, I had the privilege of working with Sigma.7 haptic device. I want to thank Micah Prendergast for helping me understand the complexities of the device. I want to express my gratitude to Corrado Pezzato and Chadi Salmi from AIRLab for providing me the simulation environment of the remote robot and helping me out when needed. Also, I want to thank Giovanni Franzese for his help with the controller and the discussion about stiffness ellipsoids.

Finally, I want to thank my brother for his insight in suggesting that the Robotics master would perfectly match my interests. He was right. I became truly fascinated by Human-Robot Interaction and I feel grateful for the opportunity to have immersed myself in research within this field. I hope the results and findings presented in this work contribute to further advancements in teleoperation-based skill transfer.

Astrid Rots
Delft, October 2023

Contents

A Shared Control Interface for Online Teleoperated Teaching of Combined High- and Low-level Skills

A.W.E. Rots
Supervised by: Dr. ir. L. Peternel

***Abstract*—We propose a novel shared control interface that enables teleoperated teaching of both high-level decision-making skills and low-level impedance modulation skills using a single haptic device. In the proposed method, high-level teaching is achieved by repurposing the haptic device to remotely modify Behaviour Trees (BTs), allowing human operators to guide decision-making. Repurposing of the haptic device is achieved by exploiting its degrees of freedom for different functionalities. Low-level skill teaching involves an impedance command interface, that is used to command endpoint stiffness by manipulating a 3D virtual stiffness ellipsoid with the haptic device. Both teaching modes are connected: a newly demonstrated low-level skill appears in the BT at a user-specified index. Control is shared between the human and the autonomous system on a high- and low-level. At the higher level, the human can change the BT online, while ongoing execution of the low-level actions within behavior tree remains uninterrupted. During low-level teaching, shared control is implemented between the robotic motion skill and human-demonstrated stiffness. To provide a proof-of-concept and demonstrate the main features of the proposed interface, we performed several experiments in a teleoperation setup operating a remote shelf-stocker robot in a supermarket environment. A predefined BT encodes high-level decisions for a pick-and-place task. The impedance command interface is evaluated in a "peg-in-hole"-like task of placing a product on a cluttered shelf. Ultimately, the proposed interface can facilitate teleoperation-based Learning from Demonstration for the transfer of both high- and low-level skills in an integrated manner.**

***Index Terms*—shared control, teleoperation, robot teaching, high-level decision-making, physical interaction skills, impedance control, Behaviour Trees.**

I. Introduction

ULTIMATELY, we strive for autonomous robots that seamlessly integrate into our daily lives to assist us or execute tasks independently. To achieve this vision, intuitive programming methods are essential, allowing and even encouraging the use of robots by non-experts.

Learning from Demonstration (LfD) offers an effective approach to realize this goal and enables humans to teach robot skills by simply demonstrating desired behaviours [1]. The LfD algorithm uses these demonstrations to generate a policy that mimics the demonstrated behaviour. LfD incorporates human cognitive capacity into the programming of robots in a natural and user-friendly manner. The demonstrations can be obtained through passive observation (robot observes human, executes a task based on sensor-derived data), kinesthetic teaching (human directly moves robot joints in the desired position), or teleoperation [2].

Teleoperation allows humans to remotely operate a robot by sending motion control commands from a haptic device to the remote robot. Devices typically used for teleoperated commanding are graphical user interfaces, joysticks or exoskeletons [3]. The advantage of teleoperation is that it provides a solution for tasks that cannot be directly executed by a human physically present in the environment, due to potentially hazardous conditions posing safety risks and/or inaccessibility of the environment. Another advantage of teleoperation-based LfD compared to passive observation and kinesthetic teaching is the physical decoupling that prevents demonstrator-induced dynamics from affecting learned policies [4]. A third advantage of teleoperation-based methods is that they facilitate shared control, due to the inherent human-in-the-loop design. Shared control enables the human operator to share the control of the teleoperated robot with an autonomous controller [5]. On the contrary, kinesthetic teaching has limited human-in-the-loop control and interaction possibilities (only joint movement can be demonstrated), preventing continuously and effectively shifting control authority between human and robot [6]. Many teleoperation-based LfD methods employ shared control during the robot execution phase to guide online teaching and improve skill transfer [6]–[14].

Literature shows that teleoperation-based LfD has been successfully implemented for the transfer of *low-level* physical interaction skills [15]. Low-level skills are the primitive movements the robot can execute, such as pushing, reaching, grasping and placing. These skills can be based on motion control [7]–[10], [16]–[19], impedance control [4], [6], [11], [12], [20]–[22] or hybrid force/motion control strategies [13], [14]. As teleoperated robots often operate in unpredictable and/or unstructured environments to perform in-contact and interaction-based tasks, such as assembly [23], surgery [24] and collaborative tasks [21], these low-level skills are vital for dealing with the associated complex dynamics.

To achieve our goal of autonomous robots, it is essential to teach robots not only low-level skills but also *high-level* decision-making skills. LfD for the learning of high-level skills has three main components. First, segmentation, which involves identifying the repeated structures within the provided demonstrations. Second, the learning of low-level skills, as previously explained. Finally, sequencing, which denotes inferring and learning the transitions between the learned low-level actions [25]. Teaching robots high-level skills greatly increases robot capabilities. The nature of interaction tasks is highly complex, often demanding anticipation instead of single reactive, primitive actions. With high-level reasoning, the robot can adapt its overall action sequence. Additionally, employing a joint approach that integrates high- and low-

level skill transfer reduces state space complexity, yielding increased planning efficiency [26]. Utilizing only low-level skills without high-level actions requires a larger number of demonstrations to effectively cover a certain behaviour or problem [27].

High-level LfD requires a model to represent the high-level decision-making. Existing methods use Behaviour Trees (BTs) [25], [28], Finite State Machines (FSMs) [29]–[31], Hierarchical Task Networks (HTNs) [32], [33] or symbolic reasoning [34]–[37]. Existing high-level LfD methods rely on kinesthetic teaching [28], [30], [31], [34]–[37] to record demonstrations, rather than teleoperation. However, given the previously mentioned advantages of teleoperation-based LfD over kinesthetic teaching-based LfD, there is a clear need to explore the extension of teleoperation-based LfD to encompass high-level skill acquisition and integrate it with existing low-level LfD. Notably, some high-level LfD methods use a GUI to demonstrate action sequences [25], [32] showing the benefit of a communication tool between human and robot to guide the learning. As teleoperated teaching through shared control has proven to be successful for low-level skill learning, we argue that shared control could also be effectively applied to guide high-level teleoperation-based LfD. Furthermore, existing high-level LfD methods only incorporate motion skills as the low-level primitive actions [25], [28]–[37]. The inclusion of low-level force and impedance skills would allow for representing increasingly complex interaction behaviours.

Addressing this gap in the field of teleoperation-based LfD, we propose and develop a novel shared control interface for facilitating online multi-modal teleoperated teaching of both high-level decision-making skills and low-level physical interaction skills. The contributions of this research are summarized as follows:

1) Teleoperated teaching through shared control of both high- and low-level skills using a single master device.
2) High-level teaching is achieved by temporarily repurposing the master device. Behaviour Trees (BTs) are used to represent high-level decision-making. Hence, high-level teaching involves remotely modifying the BT using the haptic device.
3) Inclusion of a novel 3D impedance command interface to teach stiffness modulation as the low-level skill.

These extensions are essential to facilitate a transition towards remote autonomous robots that can be programmed intuitively. Fig. 1 provides a schematic overview of the proposed teaching approach. It is crucial to emphasize that our research objective is centered around *teaching*, rather than learning. Teaching denotes the exact replication of the demonstrated action. The goal is to develop an interface that simplifies remotely commanding a robot on a high- and low-level in an integrated manner using a single master device. Ultimately, such an interface could facilitate the machine learning aspect of LfD on both levels.

The teleoperated teaching framework is based around the Force Dimension Sigma.7 as the master device. The remote robot is a Franka Emika Panda. The proposed framework and

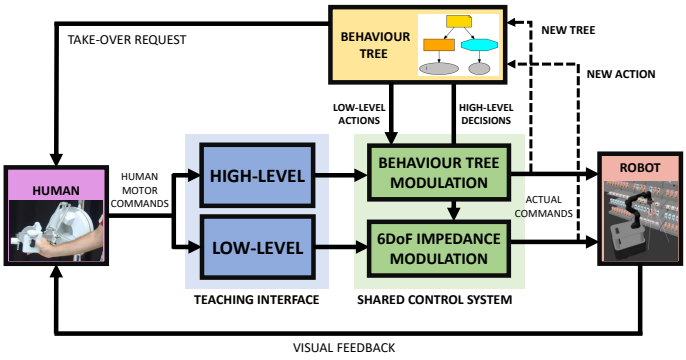


Fig. 1: The proposed teaching approach for combined high- and low-level skills with shared control between the robotic skills and human commands. Human operator uses Sigma.7 master device for teaching.

interface are evaluated in a supermarket scenario. The growing importance of retail robotics [38] highlights the potential for robots to enhance efficiency in supermarkets.

A predefined BT is used to encode pick-and-place behaviour for a robot operating in a supermarket [39]. A human operator remotely supervises the robot from a shared control[1] perspective, retaining the authority to interrupt tasks at varying levels. At the higher level, the human supervisor can modify the BT to achieve different action sequences. So either human or BT is controlling the high-level decisions. Note that, if the human takes over high-level control, the ongoing execution of the BT low-level actions remains uninterrupted. At the lower level, the operator can take control during product placement to teach variable stiffness trajectories for increased placing precision. At this level, shared control is implemented between the robotic motion skill and human-demonstrated stiffness: the human specifies the stiffness along a kinematic trajectory established by the autonomous system. These newly demonstrated low-level actions will immediately appear in the BT, connecting both levels.

Haptic devices for teleoperation are primarily designed for low-level skill transfer, i.e. to command motion (end-effector poses) [8] and/or force [11]. This makes them less suitable for teaching high-level decision-making skills, for example specifying the order of an action sequence or a condition-action relation. Additionally, operating a haptic device is generally more challenging than simple joint movements used in kinesthetic teaching. Given the already considerable difficulty in high-level LfD, researchers may opt for the easier method of kinesthetic teaching for recording demonstrations. This has hindered further progress in the field of teleoperation-based LfD. In summary, repurposing existing haptic devices to achieve high-level teaching in an intuitive manner represents a critical challenge we aim to address through our research.

While teleoperated teaching of high-level skills could theoretically be achieved using a computer mouse and/or keyboard, repurposing the haptic device offers several advantages over these conventional input devices. First, as explained earlier,

[1]We implemented *traded control*, a subset of shared control in which the shared control percentage is either fully on the human or the robot.

it is desired to combine high- and low-level teaching. This is impossible with a computer mouse, as the human operator must switch between different teaching interfaces. Switching involves cognitive changes, and maybe even requires physical movement, which is inconvenient for the human operator. Furthermore, it is time-consuming and complicates real-time teaching. The use of a single haptic device is a first step towards a paradigm in which humans can teach a complete skill set to remote robots using a portable setup, without dependence on other hardware. Finally, another advantage that underscores our design choice is that the haptic device has the ability to provide multi-dimensional inputs, which opens up a broader spectrum of teaching possibilities for complex high-level decision-making, compared to 2D mouse inputs.

We chose BTs as the high-level decision-making framework because they facilitate high-level teaching, due to their advantages in terms of modularity and reactivity compared to other frameworks [40], [41]. First, these features make BTs transparent and understandable [41] for the human operator. The human can easily grasp the structure and flow of the tree, making it straightforward to teach high-level sequences to the robot. Second, the modular nature allows for generation of a library of sub-trees. This simplifies the process of teaching by enabling the assembly of complex sequences from predefined, reusable building blocks. Third, the reactiveness makes BTs suitable for unpredictable environments. This aligns perfectly with our objective of achieving teleoperated teaching, considering that teleoperation setups are typically employed in such environments. Finally, the reactivity of BTs not only enables quick responses to environmental changes but also allows the human operator to take control and teach a new sequence or modify an existing one on-the-fly.

II. Method Design

The main objective of the proposed system is to enable teleoperated teaching of both high-level decision-making and low-level impedance modulation skills using a single haptic device. The proposed method is based on repurposing the haptic device, i.e. exploiting its different degrees of freedom (DoF) for different functionalities. To get these DoF, we practically employed Sigma.7 haptic device. The human operator can manipulate the endpoint of the haptic device in 6 DoF and control a finger-operated gripper. The gripper can be fully opened or closed by the user's finger movements, and a button click function is emulated by rapidly moving the gripper back and forth. Fig. 2 provides a system overview with the main software blocks, signals and apparatus. Key elements are the manipulating of the BT with the repurposed haptic device and modulation of the stiffness ellipsoid in 3D, both highlighted in yellow blocks. To enhance operator immersion, we implemented a graphical user interface (GUI) that provides visual feedback for both teaching modes. Section II-A describes requirements, while Sections II-B and II-C elaborate on both teaching modes separately.

A. Design Requirements

The following requirements served as the basis for the interface development:

1) **On-the-fly-usage:** The human operator must be able to use the shared control interface on-the-fly. Teaching and then updating of the BT must be achieved in an online manner.
2) **Single haptic device:** Teaching of both the high- and low-level skills is achieved with the haptic device alone, i.e. no additional devices are required for successful use of the interface.
3) **Multi-axis impedance commanding:** The low-level teaching mode of the interface should allow impedance commanding in 6 DoF.
4) **Visual feedback:** The interface should provide visual feedback to the human operator regarding the current robot state and the BT.
5) **Multi-modality:** The interface should allow the human operator to make a wide range of changes to the BT, to address a wide variety of unforeseen situations.
   a) High-level teaching:
      - Changing the structure of the tree
      - Adding nodes
   b) Low-level teaching:
      - Demonstrating a new skill
6) **Shared control:** The interface must enable teaching in a shared control manner.
   a) Human operator detects a problem and overtakes control
   b) Robot detects own insufficiency and asks the human operator for a contingency plan

B. High-level Teaching

High-level teaching involves demonstrating decision-making to the robot. Among the three types of high-level decision-making (segmentation, sequencing, and conditioning) our focus is on sequencing: determining the order in which a series of actions should be executed to achieve a particular goal. When teaching sequencing skills, the human operator demonstrates the correct sequence of actions to achieve a desired outcome. We employed BTs as the framework for high-level decision-making. In our high-level teaching mode, the human operator can modify the sequence of actions described by the BT in an online manner using the haptic device. This BT was created beforehand using a set of predefined skills.

To enable the human operator to command a new action sequence and manipulate the structure of the BT effectively, a set of functionalities and operations is essential. This includes the ability to remove existing nodes, add new ones, or insert nodes at specific positions within the tree. We repurposed the haptic device to provide these functionalities, capitalizing on its DoF. The functionalities are presented to the human operator on the GUI. Two translational DoF are employed for selecting a node and specifying the desired operation (removing, adding, prepending or inserting). One rotational DoF is used for selecting the index at which a new node must be inserted. Finally, the emulated button click action, executed with the finger-operated gripper, is used for implementing the

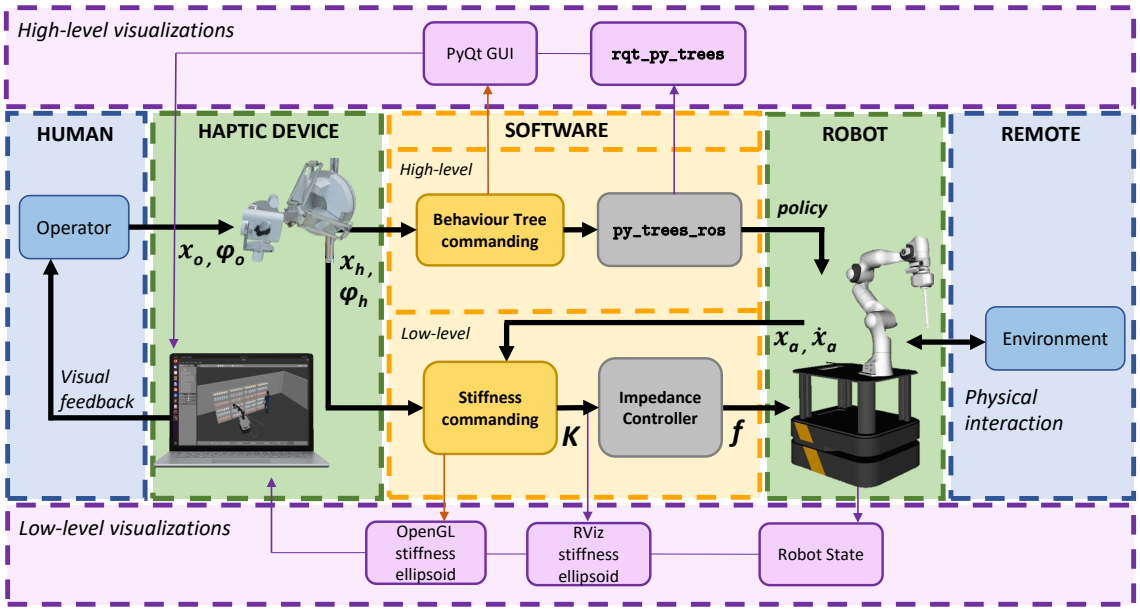


Fig. 2: System overview depicting core software blocks, signals and apparatus. The human operator and remote environment (blue components) interact with the haptic device and remote robot (green components), respectively. The human operator provides 3D position $\boldsymbol{x}_o$, orientation ϕ_o and emulated button click inputs to the haptic device. The haptic device sends position $\boldsymbol{x}_h$ and orientation ϕ_h to the GUI. The yellow section highlights the software structure for both high- and low-level teaching modes. For each teaching mode, the visualization elements are shown in the purple blocks, and connected to their corresponding software part. The yellow blocks highlight the two fundamental elements of the teleoperated teaching interface.

selected changes within the BT. Once the human operator finishes teaching, he/she can fully open the finger-operated gripper to return to the main window. The BT is updated at runtime and any changes will immediately appear in the visualization tool of the tree. The upper part of Fig. 2 shows the structure of the high-level teaching mode, and how the software blocks are connected to relevant visualizations.

C. Low-level Teaching

The low-level teaching mode aims to teach stiffness modulation through a novel impedance command interface. The human operator supervises BT execution and can take control during product placement to command a variable stiffness trajectory. The demonstrated stiffness is recorded to allow for subsequent reuse of this skill within the BT. Robot endpoint stiffness can be represented as an ellipsoid defined by three parameters: size (the area of the planar ellipses), shape (ratio of principle axes) and orientation (direction of major axis) [42]. In [43] a method was proposed that achieves independent control of all these aspects using a virtual stiffness ellipsoid on a touchscreen device. We have built upon the interface proposed in [43], by implementing a virtual stiffness ellipsoid command interface that repurposes the haptic device to manipulate the stiffness ellipsoid. Our method extends the existing state-of-the-art in three ways: 1) 3D representation of the stiffness ellipsoid, 2) changing the stiffness ellipsoid across all 6 DoF, and 3) increased intuitiveness by using spatial inputs from a haptic device instead of a 2D touchscreen.

The virtual ellipsoid is visualized on a GUI and controlled with the haptic device. The GUI is split into four sections, each showing a different view of the stiffness ellipsoid: three-dimensional, X-Y plane, Y-Z plane and X-Z plane. The stiffness ellipsoid is represented in the robot base frame (which is aligned with the world frame). Three translational DoF of the haptic device are employed for scaling the size and changing the shape of the stiffness ellipsoid along its principal axes. Three rotational DoF are used for rotating the stiffness ellipsoid along the X, Y, and Z axes. Finally, the emulated button click action, executed with the finger-operated gripper, serves a dual purpose: selecting the principal axis to be controlled and starting/stopping the recording of a stiffness trajectory.

Singular value decomposition (SVD) shows that multiplying the unit sphere by any mxn matrix M, yields a stretched and rotated shape, forming an ellipsoid. The goal of the low-level teaching mode is to construct a symmetric positive definite (SPD) stiffness matrix $\boldsymbol{K}$. For SPD matrices, singular value decomposition is equal to the eigenvalue decomposition (EVD). Because endpoint stiffness can be represented as an ellipsoid, we can exploit the properties of ellipsoids to determine the eigenvalues $\boldsymbol{\Sigma}$ and eigenvectors $\boldsymbol{V}$ and then use eigendecomposition to construct the remote robot's endpoint stiffness matrix according to

$$\boldsymbol{K}_T^{new} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{1}$$

where $\boldsymbol{K}_T^{new} \in \mathbb{R}^{3\times3}$ is the new translational part of the stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{6\times6}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{V} \in \mathbb{R}^{3\times3}$ denote the eigenvalues and eigenvectors, respectively.

The human operator can directly set the lengths of each of the semi-axes using the translational DoF of the haptic

device. The lengths of the semi-axes of the stiffness ellipsoid correspond to the eigenvalues of $\boldsymbol{K}$[2]. The semi-major axis represents the direction along which the end-effector is stiffest. Note that the eigenvalue matrix $\boldsymbol{\Sigma}$ is pre-multiplied by a diagonal matrix to obtain the right units, converting from pixels [px] to robot stiffness [N/m].

The human operator can change the roll ϕ, pitch θ and yaw ψ angles of the ellipsoid in the robot base frame using the rotational DoF of the haptic device. These Euler angles denote rotations around the fixed axes X, Y, and Z respectively. The three consecutive rotations form the orthogonal rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$,

$$\boldsymbol{R} = \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{R_x}(\phi), \tag{2}$$

which represents the directions of the semi-axes in the robot base frame. These direction vectors of the semi-axes of the stiffness ellipsoid correspond to eigenvectors of the stiffness matrix, making $\boldsymbol{R}$ equivalent to the eigenvector matrix $\boldsymbol{V}$.

The full robot stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{6\times6}$ is given by

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_T^{new} & 0 \\ 0 & \boldsymbol{K}_R^{new} \end{bmatrix}, \tag{3}$$

The proposed low-level teaching mode does not implement a method for commanding the magnitude of rotational stiffness component $\boldsymbol{K}_R^{new} \in \mathbb{R}^{3\times3}$, but the rotation $\boldsymbol{R}$ must also be applied to the rotational axes through

$$\boldsymbol{K}_R^{new} = \boldsymbol{V}\boldsymbol{K}_R\boldsymbol{V}^T, \tag{4}$$

where $\boldsymbol{K}_R^{new} \in \mathbb{R}^{3\times3}$ is the new rotational part of the stiffness matrix and $\boldsymbol{K}_R \in \mathbb{R}^{3\times3}$ is the predefined rotational stiffness.

To control the remote robot with the derived stiffness matrix $\boldsymbol{K}$ we employed Cartesian impedance control (see the lower part of the yellow section in Fig. 2), which models the robot's physical interaction at the end-effector a mass-spring-damper mechanism. Cartesian impedance control is defined as

$$\boldsymbol{f} = \boldsymbol{K}(\boldsymbol{x}_r - \boldsymbol{x}_a) + \boldsymbol{D}(\dot{\boldsymbol{x}}_r - \dot{\boldsymbol{x}}_a), \tag{5}$$

where $\boldsymbol{f} \in \mathbb{R}^6$ is the interaction force acting from the remote robot endpoint on the environment, $\boldsymbol{x}_r \in \mathbb{R}^6$ and $\boldsymbol{x}_a \in \mathbb{R}^6$ are the reference (i.e., desired) and actual endpoint pose, respectively, $\boldsymbol{K} \in \mathbb{R}^{6\times6}$ is the Cartesian stiffness matrix continuously obtained from 3, and $\boldsymbol{D} \in \mathbb{R}^{6\times6}$ is the positive definite Cartesian damping matrix. When low-level teaching mode is chosen, the human teaches the stiffness along a predefined path $\boldsymbol{x}_r$. This motion was generated by specifying a set of waypoints and no further emphasis is placed on how these kinematic trajectories are established.

The damping matrix $\boldsymbol{D}$ is calculated at each time-step based on the commanded stiffness matrix and rotated to match its orientation. To achieve this, we need the eigendecomposition of the new stiffness matrix

$$\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{K}_0\boldsymbol{Q}^T, \tag{6}$$

where $\boldsymbol{Q} \in \mathbb{R}^{6\times6}$ and $\boldsymbol{K}_0 \in \mathbb{R}^{6\times6}$ are eigenvectors and eigenvalues. We can now use $\boldsymbol{D} = 2\boldsymbol{\xi}\sqrt{\boldsymbol{K}}$ with damping ratio ξ set to 0.7 for a critically damped system [45] to determine the damping matrix $\boldsymbol{D}$ through

$$\boldsymbol{D} = 2\boldsymbol{Q}\boldsymbol{D}_\xi\sqrt{\boldsymbol{K}_0}\boldsymbol{Q}^T \tag{7}$$

where the diagonal matrix $D_\xi \in \mathbb{R}^{6\times6}$ contains the damping ratios for each DoF.

The commanded stiffness matrix $\boldsymbol{K}$ is stored at each timestep. Once the teaching is finished, a new node is added to the BT skill library, which achieves the product placement in the shelf by utilizing the demonstrated stiffness.

III. Experiments & Results

To demonstrate the proposed shared control interface for teleoperated teaching of combined high- and low-level skills using a single haptic device, we conducted proof-of-concept experiments. The teleoperation setup consisted of a Force Dimension Sigma.7 haptic device and a simulated Franka Emika Panda robotic arm mounted on a Clearpath mobile base. The Gazebo simulation of the robot and the supermarket environment were provided by AIRLab Delft[3]. Section III-A describes how the haptic device was repurposed and elaborates on the software and experimental setup used to implement the high- and low-level teaching modes. The proof-of-concept experiments involved both teaching modes. The goal of the high-level task (Section III-B) was to change the sequence of the BT by removing and inserting nodes. The low-level task is product placement, in which a variable stiffness trajectory is necessary to ensure both safe interaction and precise execution. To show how the low-level teaching interface can be used to change size, shape and orientation of the stiffness ellipsoid, two separate product placement tasks are evaluated: from a frontal shelf-facing position (Section III-C), and from a 45-degree angle relative to the shelf (Section III-D). In Section III-E, we illustrate how both teaching modes can be combined through the interface: the newly demonstrated product-placing skill is added to the skill library and can be inserted in the BT using the haptic device.

A. Experimental Setup

The interface is developed using the ROS software framework. The Sigma.7 has a gripper, three translational and three rotational DoF. Its force feedback functionality was not utilized in this study. Repurposing of the haptic device is achieved by exploiting the incoming data from Sigma.7 through the ROS publisher/subscriber communication structure. The graphical user interface (GUI) that is provided to the human operator for visual feedback and guidance during both teaching modes is created with PyQt. The main window of the GUI has the option to select either the high- or low-level teaching mode. Fig. 3 shows the overview and internal structure of the GUI. Fig. 4 visualizes the repurposing of the haptic device for both high- and low-level teaching modes on a conceptual level. Appendix A provides a detailed visualization and discussion of the ROS communication structure.

[2]The eigenvalues are equal to the inverse of the semi-axes lengths squared, however, this yields the *isopotential* stiffness ellipse which is not a direct representation of endpoint stiffness values [42], [44]. This research therefore uses the semi-axes lengths directly.

[3]https://icai.ai/airlab-delft/

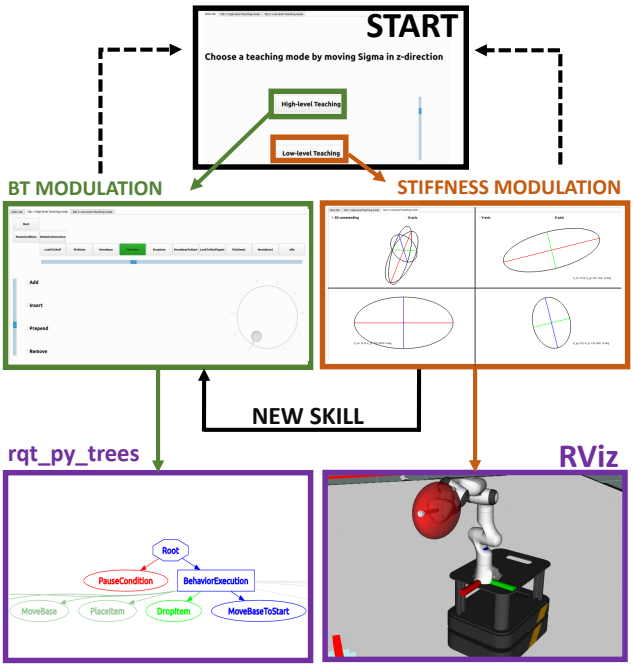


Fig. 3: Overview of the internal structure of the graphical user interface (GUI), used to guide the teaching. The human operator navigates the GUI using the Sigma.7 haptic device. The human selects high- or low-level teaching mode. The high-level window (green block) has artificial buttons for each BT node. If a button is selected, the human operator can choose to add/prepend/remove it or insert the same node at another index. The BT is updated at runtime and visualized with `rqt_py_trees` (left purple block). The low-level window (red block) shows a 3D visualization of the endpoint stiffness ellipsoid that the human operator can modulate in 6 DoF using the haptic device. A visualization of the commanded ellipsoid is shown in red in RViz (right purple block). If the user finishes low-level teaching, a new skill is added to the high-level GUI.

If the high-level teaching mode is selected, the human operator sees the window highlighted green in Fig. 3 and Fig. 4. The interface displays buttons that directly correspond to the nodes in the BT. The BT is modelled with `py_trees_ros` and visualized with `rqt_py_trees` (see purple block in Fig. 3). The buttons match the nodes in the tree, both in name and structure. The BT is set up beforehand using a skill library. This skill library contains simple actions necessary for achieving pick-and-place behaviour in a supermarket, like "LookToShelf", "MoveBase", "PickItem", "DropItem". These low-level actions were created using the `MoveIt` motion planning framework for ROS. During these actions, the robot arm is position controlled. Appendix B further elaborates on the predefined BT. The high-level teaching window of the GUI gives the human operator the capability to manipulate the sequence of the initial BT, by navigating the GUI using the haptic device as explained in Section II-B and shown in Fig. 4.

If the low-level teaching mode is selected, the human operator sees the window that is highlighted red in Fig. 3

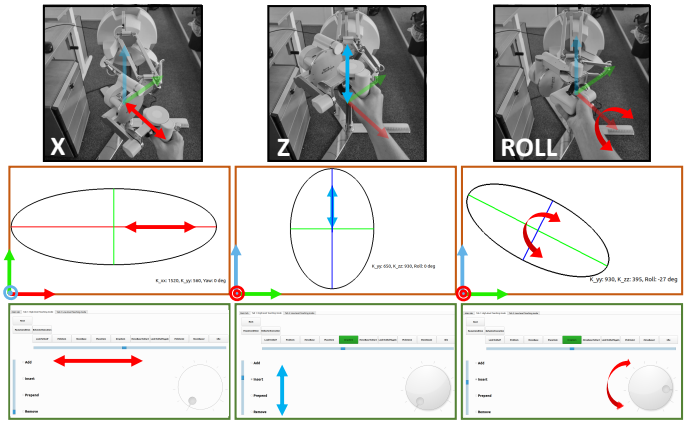


Fig. 4: Repurposing the haptic device for both teaching modes. The top layer shows the haptic device and its reference frame. In the low-level teaching mode (red GUI sections), the user can move the haptic device back and forth along the translational axes to change the length of the ellipsoid principal axes. The user can rotate the haptic device around its rotational axes to change the orientation of the ellipsoid in the robot base frame. In the high-level teaching mode (green GUI sections), the user can move the haptic device back and forth along the translational axes to select nodes and operations to perform. The user can rotate the haptic device around the X-rotational axis to specify the index for insertion in the BT.

and Fig. 4. The low-level teaching window of the GUI gives the human operator the capability to control robot endpoint stiffness in all 6 DoF, using the haptic device as explained in Section II-C and shown in Fig. 4. When the low-level teaching is started, the BT execution is paused through the blackboard[4]. Furthermore, the ROS controller switching functionality is used to stop the position controller used by `MoveIt` and start the Cartesian impedance controller. We used and adapted the Cartesian impedance controller for Franka Emika Panda from [46].

The low-level teaching window provides four options to the human operator for commanding the DoF: commanding all 6 DoF simultaneously, commanding X-axis length and roll angle, Y-axis length and pitch angle, Z-axis length and yaw angle. The human operator can select the desired axes with the emulated button click action. Every time the button is clicked, the next axis is selected. This way, the human operator can sequentially set stiffness in the X, Y, and Z axis, or he/she can choose to control stiffness in all directions at once. The GUI displays the commanded stiffness values for each principal axis along with its corresponding angle around that axis. This feature helps users understand the order of magnitude of the stiffness commands.

The emulated button click action is also used to start/stop recording of the stiffness trajectory. When recording starts, a predefined kinematic reference trajectory ($\boldsymbol{x}_r$ in equation 5) to a designated spot in the shelf is sent to the Cartesian impedance controller. No assumptions are made on how the

[4]The blackboard is a key/value storage system that can be accessed by all nodes in the BT.

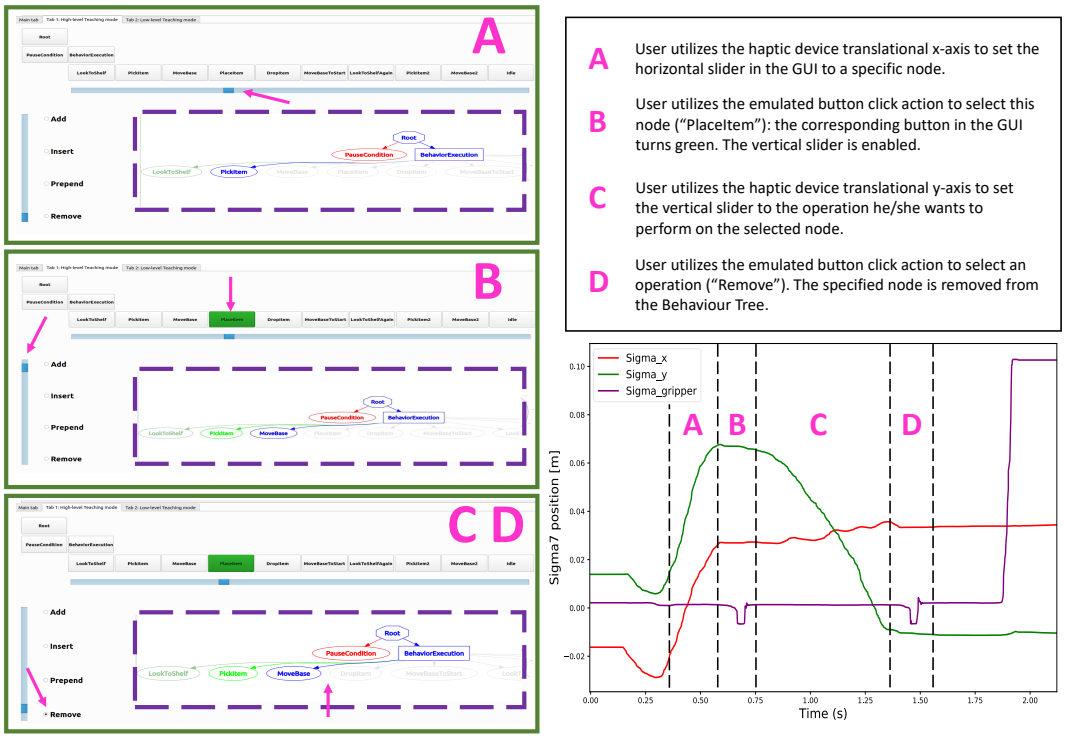


Fig. 5: Results of the high-level task: the human operator uses the haptic device to change the BT online. The left column shows consecutive screenshots of the GUI (green sections) and the visualization of the current BT (dotted purple sections). The graph in the bottom right shows the haptic device inputs (translational X and Y and finger-operated gripper) used to navigate the GUI and change the BT. Four consecutive phases (A,B,C,D) in the haptic device movements can be distinguished, which are elaborated on in the legend.

kinematic trajectory is established beforehand: for example, it could be generated remotely with the same haptic device used for stiffness demonstration.

The 3D visualization of the stiffness ellipsoid is created with OpenGL bindings for PyQt, that enable simple graphics rendering inside the GUI window. During teaching, the human operator can see a visualization in RViz of the endpoint stiffness ellipsoid that he/she is commanding. If the human operator finishes demonstrating and stops the recording, the new skill is added to the skill library. The human operator can then return to the high-level window and use the haptic device to insert the new product-placing skill, that uses the demonstrated stiffness, in the BT at a specific index. Returning to the main window is indicated by the dotted arrows in Fig. 3 and is accomplished by an additional repurposing of the haptic device. If the human operator uses his/her finger to move the gripper to its fully opened position, the main window is opened.

The haptic device inputs were scaled to match the maximum pixel width and height of the screen. This scaling was applied to ensure that the ellipsoid renderings spanned the entire screen. The obtained eigenvalue matrix $\mathbf{\Sigma}$ is premultiplied by a diagonal matrix with values of 5 [N/(m·px)], to obtain the right stiffness units and set the maximum translational stiffness to 3000 N/m in each DoF. The rotational stiffness matrix $\boldsymbol{K}_R$ was preset to 30 Nm/rad in each rotational DoF.

B. High-level Task: Demonstrating a New Action Sequence

This Section covers the high-level teaching part of the proof-of-concept experiments and shows how our proposed interface facilitates the online modification of the BT through the repurposed haptic device. The results of the task are shown in Fig. 5, which shows how the human operator used the haptic device to navigate the GUI to change the BT. The haptic device movements can be categorized into four consecutive phases.

The translational X-axis of the haptic device is used for selecting a node, this is reflected by the increasing red graph (Phase A). The Sigma.7 haptic device is difficult to operate in a single DoF only, as it is very sensitive to small movements; this is reflected by the translational Y-axis position that increases at the same time (green graph). In this phase, we do not want to utilize the translational Y-axis yet, so the vertical slider remains disabled until a node is selected. Once the position of the slider is underneath the node to be selected (in this case "PlaceItem"), which is reflected by the red graph remaining constant, the user utilizes the emulated button click action with the finger-operated gripper. This is indicated by the sudden drop in the purple graph (Phase B). The vertical slider is now enabled. The GUI starts listening to the translational Y-axis data. The horizontal slider is disabled and the GUI stops listening to translational X-axis data. The user moves the slider to the operation he/she wants to perform on the selected node: this is reflected by the green graph decreasing (Phase C). Once

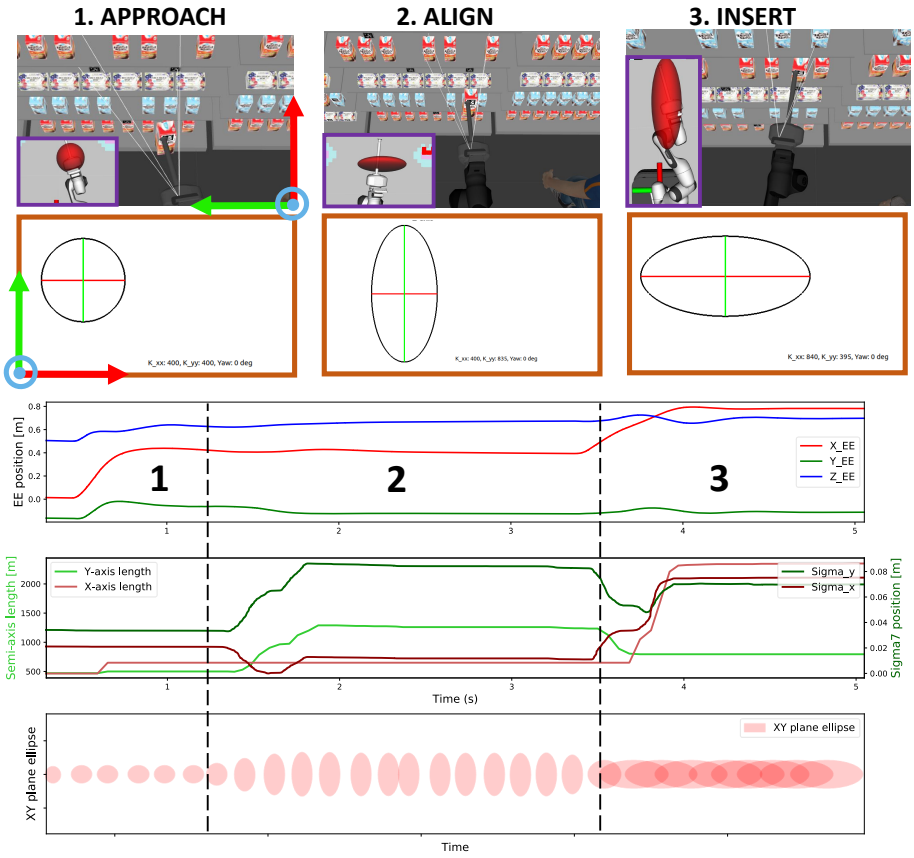


Fig. 6: Result of the product placement task from a position perpendicular to the shelf, highlighting how stiffness in the X-Y plane was varied for each phase (approach, align, insert). The top layer shows three consecutive screenshots of the robot end-effector during product placement. In the corner, the RViz visualization of the commanded stiffness ellipsoid is depicted. The robot base frame is indicated. The user controlled the haptic device translational DoF to modify the stiffness ellipsoid through the GUI as shown in the second layer. The user controls stiffness in the robot base frame. For clarity, only the X-Y plane section of the GUI is displayed. The third layer shows the graph of the actual Cartesian end-effector position during placement. The fourth layer shows the Sigma.7 haptic device movements (dark coloured) and the corresponding principal axes lengths of the ellipsoid (light coloured), i.e. the principal stiffness components in the robot base frame. The fifth layer shows the X-Y plane of the commanded stiffness ellipsoid over time.

the user has the haptic device steady (in this case at the "Remove" operation), he/she uses the emulated button click action again (Phase D). The ROS service/client structure is utilized to send the request to the BT (see Appendix A). The requested node is removed from the tree (see purple dotted block). After phase D, we see an increase in the gripper angle to its maximum value (i.e., fully opened position). We repurposed this DoF of the haptic device for navigating back to the main window to start low-level teaching.

C. Low-level Task 1: Changing Size and Shape of the Stiffness Ellipsoid

This Section describes how the ellipsoid interface is used to command stiffness in multiple axes during the low-level skill of placing a product on the supermarket shelf, which resembles the classic "peg-in-hole" task. The results are shown in Fig. 6. During the placing, we can distinguish three different phases in terms of the desired size and shape of the stiffness ellipsoid: 1) approaching the designated drop-off spot in the shelf, 2) alignment with the drop-off location and 3) inserting the product. In the experiment, the mobile base of the robot was positioned perpendicular to the shelf, therefore the stiffness ellipsoid did not have to be rotated and the stiffness matrix $\boldsymbol{K}$ was diagonal. The direction of alignment and insertion correspond with the robot base frame Y-axis and X-axis, respectively. Endpoint stiffness in the Z direction is kept constant at a relatively high value, to account for the unknown mass of the product. Once the low-level teaching mode is started, the user can preset the stiffness ellipsoid to the size, shape and orientation desired for the start of the task. Then the emulated button click action is utilized, which starts playback of the predefined reference kinematic trajectory, $\boldsymbol{x}_r$. Fig. 6 shows how the human operator changes the X and Y principal axes of the stiffness ellipsoid during each phase using the corresponding translational DoF of the haptic device.

During approach (phase 1), the robot must be compliant to ensure a safe interaction with the unpredictable supermarket environment, where customers might be in close proximity.

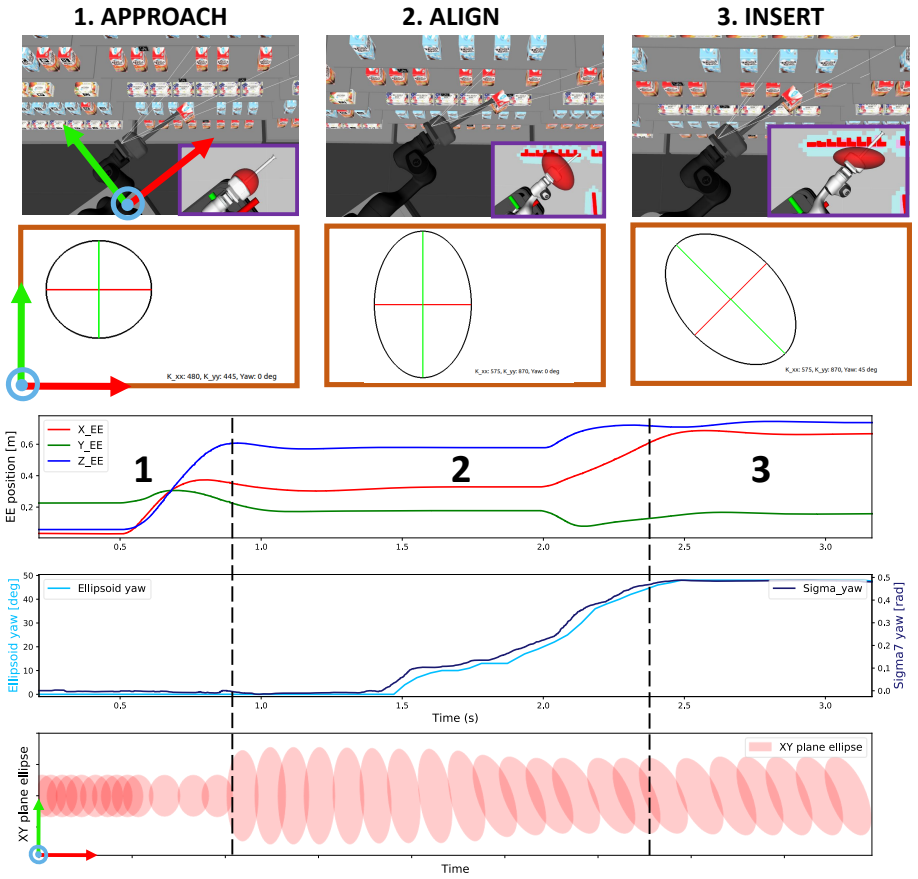


Fig. 7: Result of the product placement task from a 45-degree angle relative to the shelf. The robot base frame is rotated with respect to the shelf as depicted in the top layer. The top layer shows the three phases of product placement. The second layer shows the corresponding GUI windows. The first graph is the actual end-effector position. The second graph shows how the haptic device rotational Z-axis is used to rotate the stiffness ellipsoid around its principal Z-axis. The last graph shows how the ellipsoid changes in size, shape and orientation during product placement.

The human operator therefore commands equally low stiffness in X and Y. The X-Y intersection of the ellipsoid becomes circular.

During alignment (phase 2), it is important to reject perturbations in Y to ensure precise positioning in front of the designated drop-off location. The human operator selects the option to command the Y-axis with the emulated button click action. The user moves the haptic device in the Y direction (dark green line in the second graph) to increase the Y-axis length (light green line) of the ellipsoid depicted in the GUI. As the human operator moves the haptic device in Y, the position in X (dark red line) is also changed, however, this does not affect the X-axis length of the ellipsoid (light red line), because the user selected Y-axis commanding for this stage. Stiffness in X direction remains the same as during approach, to prevent high contact forces.

During insertion (phase 3), a "peg-in-the-hole" strategy [4], [47] is followed. Stiffness in the Y-axis is decreased slightly to make the robot more compliant in this direction, which is needed to prevent the other products from being pushed over at the slightest misalignment. Then the user selects X-axis commanding to increase stiffness along the X-axis. This is reflected in the second graph: the X-axis length of the ellipsoid increases according to the haptic device translational X position. This increased rigidity in the insertion direction forces the product to be "pushed" into the right spot.

From the second graph in Fig. 6 it follows that a delay is present between the haptic device movement and the ellipsoid principal axes size: the latter (lighter coloured lines) lag behind. The delay is caused by the update and refresh rate of the GUI being considerably lower than the Sigma.7 node refresh rate. The third graph in Fig. 6 shows how the X-Y plane of the ellipsoid changes in size and shape during the product placement: small and circular for compliant behaviour (1), principal Y-axis becomes the semi-major axis of the ellipse for precise alignment (2) and principal X-axis becomes the semi-major axis of the ellipse to force the product to the designated spot (3).

D. Low-level Task 2: Changing Orientation of the Stiffness Ellipsoid

To demonstrate the functionality of the interface to change the orientation of the ellipsoid in the robot base frame, a second low-level task was executed: product placement from

a 45-degree angle with respect to the shelf. This scenario could become relevant if a customer obstructs the robot from positioning itself directly in front of the shelf. The results are shown in Fig. 7.

During approach, (phase 1), the same compliant behaviour in both X and Y direction is necessary for safe interaction. The orientation of the ellipse is still aligned with the robot's base frame.

During alignment (phase 2), the user increases stiffness in the Y direction of the robot base frame for perturbation rejection. The user gradually starts rotating the ellipsoid around its principal Z-axis ($t = 1.5s$). This rotation is visualized in the second graph of Fig. 7, where the user manipulates the haptic device's rotational Z-axis (dark blue line), resulting in a corresponding increase in the ellipsoid's yaw angle (light blue line). The user rotates the ellipsoid until its principal Y-axis coincides with the required direction of alignment to place the product on the shelf (45 degrees). The bottom graph illustrates the rotational adjustment of the ellipsoid. At last, the Y-axis length is slightly decreased to mitigate the risk of high contact forces during the impending insertion phase.

During insertion (phase 3), the orientation of the ellipsoid, now properly aligned, is kept constant. Notably, the "peg-in-the-hole" strategy, i.e. increased stiffness in the insertion direction to force the product into the right spot, was intentionally omitted during this experiment. The reason for this omission lies in the specific task context. Due to the robot's angled positioning, pushing the product forcefully in the insertion direction would risk displacing other nearby products. The angled configuration limits how far the product can be inserted into the shelf, making the conventional strategy less relevant in this context.

We took initial steps of using the proposed interface for low-level skill learning, by using Dynamic Movement Primitives (DMPs) to encode the stiffness trajectory for product placement at an angle relative to the shelf, see Appendix C for the preliminary results.

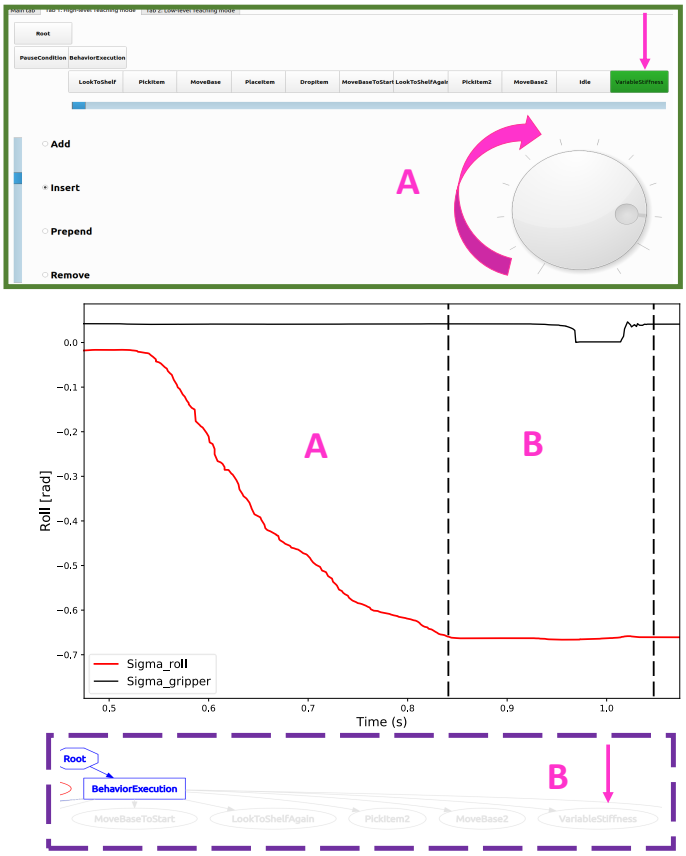


Fig. 8: Overview of how the newly demonstrated low-level skill can be implemented in the BT, so the robot can employ it for later use. After low-level teaching, a new skill appears in the high-level window (green section). The user can select this skill and operation using the haptic device. In phase A, the user uses the haptic device rotational Z-axis to select the index at which the skill must be inserted. In phase B, the user utilizes the emulated button click action to implement the insertion. The new skill is added to the tree, as seen in the purple dotted block.

E. Skill Integration in the Behaviour Tree

The ultimate goal of demonstrating a new skill is for the robot to use it in future tasks. When the low-level teaching ends, a new button is added in the high-level GUI. This button encodes the previously demonstrated variable stiffness trajectory. This is indicated by the pink arrow in the high-level window (green section) in Fig. 8. Using the haptic device, the user can choose to insert this new skill. This is similar to the process described in Section III-B. Once the "Insert" option is selected, a dial mechanism in the GUI is enabled to allow the user to specify at which position in the BT the skill must be inserted. The user controls the dial with the rotational X-axis of the haptic device. This is highlighted by phase A. Once the index is specified, the user utilizes the emulated button click action executed via the finger-operated gripper (phase B), and the node is inserted in the BT (purple dotted block).

This shows how the proposed interface can connect both teaching modes. From a high-level perspective, the human supervises BT execution and decides a new skill is needed. First, the high-level sequence must be changed, and then the low-level skill is demonstrated. Using the high-level teaching mode, the user can integrate the new skill in the BT for future use. It is important to highlight that the skill replacement is tackled from a high-level perspective: the human operator uses the high-level teaching mode to make decisions about the sequence of actions, reflecting a higher level of understanding.

IV. Discussion

The results of the proof-of-concept experiments show that the proposed interface can be used for teleoperated teaching of both high- and low-level skills. The main advantage of the proposed method is the use of a single haptic device for both high- and low-level teaching. This is important because existing high-level Learning from Demonstration (LfD) methods have refrained from using teleoperation to record demonstrations, due to the fact that state-of-the-art haptic devices are typically designed for low-level skill transfer. We showed that it is possible to repurpose the haptic device, and transformed it

into a tool to change the robot behaviour at a high level. This marks an important advancement in the field of field of teleoperation-based LfD. Our interface enables high-level skill teaching through teleoperation and retains the option to teach low-level stiffness modulation when needed.

All of the design requirements outlined in Section II-A are successfully implemented, except the shared control meta mode in which the robot seeks a contingency plan from the human operator. Equipping the robot with such an autonomous insufficiency-detection mechanism can only be achieved in a meaningful way if the BT skill library can be extended online through learning.

The repurposing of the haptic device is achieved through a GUI. Some existing high-level LfD methods [25], [32] have used a keyboard and computer mouse-controlled GUI for high-level skill transfer, however, these methods were restricted to offline learning and the GUIs were tailored to the task at hand. An advantage of our interface is that it facilitates online teaching, as both the high-level and low-level demonstrations are reflected in the robot behaviour in real-time. Additionally, the proposed method can accommodate any predefined BT. The buttons in the GUI are generated based on this initial BT, allowing the user to remotely modulate any BT on-the-fly.

One of the main advantages of a single haptic device is that this eliminates the dependency on other hardware. However, the inclusion of a GUI to achieve the repurposing of the haptic device could potentially compromise this essential advantage. In the process of repurposing the haptic device, the most significant challenge we faced was effectively allocating specific DoF for distinct actions while data was continuously coming in. The use of a GUI provided an elegant solution: certain parts of the GUI can be (de)activated, making it possible to "ignore" some of the incoming data when it is not needed. The GUI could be adapted to run on a portable device like a tablet or even a VR headset, maintaining the desired flexibility of the setup. Furthermore, we argue that a GUI might be crucial in creating an intuitive high-level teaching mode for the human operator. To demonstrate high-level decision-making, a communication tool like the GUI is necessary to prevent the process from becoming too abstract.

With the proposed method we have demonstrated the haptic device can effectively be repurposed for high-level teaching, by exploiting two translational and one rotational DoF. This can easily be extended to leverage other DoF as well. For example, additional DoF of the haptic device could be used to integrate the teaching of conditioning skills, so the human operator can modify the pre-and post-conditions that guide BT ticking. Furthermore, specific DoF could be utilized to change the type of nodes within the tree, such as transforming a *sequence* node into a *fallback* node. A potential limitation of the high-level teaching mode is that its current design only allows for the leaf nodes of the BT to be modified. To give the user control over the entire BT, the slider mechanism to select nodes (as explained in Section III-B) should be extended to the higher hierarchical layers of the tree.

With the low-level teaching mode, we successfully achieved independent control over all aspects of the stiffness ellipsoid. However, the practical implementation posed challenges for the human operator due to the high sensitivity of the haptic device. We took initial steps to mitigate these effects by implementing the feature of cycling through the different control options with the emulated button click. This allows for decoupled control of the principal axes lengths. However, principal axis length and rotation around this axis are still coupled: when changing the axis length, this unavoidably leads to slight changes in orientation. Promising solutions to further mitigate the effects of the haptic device sensitivity could be to implement virtual fixtures that constrain the human arm motion along the haptic device coordinate axes or to scale the inputs that are sent to the virtual ellipsoid interface.

The proposed method enables teleoperated teaching of high-level sequencing and low-level stiffness modulation skills in an integrated manner using a single haptic device. The scope of this research was to introduce the novel method and perform proof-of-concept experiments. In future work, we will conduct a human factors study to examine the usability and cognitive load of the proposed interface. Regarding usability, we want to investigate whether the proposed interface can intuitively and effectively be used in a real-world supermarket scenario. In terms of cognitive load, the proposed method must be compared to conventional mouse/keyboard input devices to confirm whether our proposed use of a single haptic device decreases workload. This user study will guide potential improvements for redesign to enhance the value of the system for human operators.

To further improve the functionality of the proposed method, we have identified two important possible extensions. First, the proposed interface allows teleoperation-based LfD methods to be extended to high-level skill transfer. Future work should focus on how machine learning algorithms can use the demonstrated sequence changes to learn a policy for autonomous BT adaptation. Second, the control approach should be extended to shared control. The interface is currently based on traded control, in which the human enters the control loop temporarily. However, as the current capabilities of autonomous robots still often require quick and timely human intervention, it is necessary to keep the human in the loop continuously. Another motivation for improving the human-in-the-loop design lies in the supervisory role of the human operator within the proposed system: the human operator must constantly supervise BT execution. In such supervisory tasks a human-in-the-loop design is especially important for maintaining situational awareness [48]. Therefore, future research on teleoperation-based high-level LfD should explore shared control mechanisms that gradually shift the control percentage between the human and the robot, for example by taking into account the robot's confidence in its learned policy. This way, the human operator can engage in collaborative high-level decision-making with the robot through the haptic device.

V. Conclusion

This research proposed a shared control interface for teleoperated teaching of combined high-level sequencing skills and low-level stiffness modulation using a single haptic device. High-level teaching is achieved by repurposing the haptic

device through a graphical user interface, capitalizing on its degrees of freedom. Low-level teaching involves a novel virtual ellipsoid interface that allows for commanding stiffness in 6 DoF using the haptic device. This research addresses a limitation in teleoperation-based Learning from Demonstration (LfD) methods, where the absence of an intuitive device for demonstrating decision-making has complicated high-level skill learning. By showing the feasibility of teleoperated teaching of high- and low-level skills using a single haptic device through proof-of-concept experiments, we have laid the foundation for advancing teleoperation-based LfD to incorporate high-level skill transfer.

REFERENCES

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration", *Springer Handbook of Robotics*, pp. 1371–1394, 2008.

[2] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[3] S. Calinon, "Learning from Demonstration (Programming by Demonstration)", *Encyclopedia of Robotics*, Springer Berlin Heidelberg, pp 1-8, 2018.

[4] L. Peternel, T. Petric, and J. Babič, "Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation," *Autonomous Robots*, vol. 42, pp. 1–17, 01 2018.

[5] G. Li, Q. Li, C. Yang, Y. Su, Z. Yuan and X. Wu, "The Classification and New Trends of Shared Control Strategies in Telerobotic Systems: A Survey", *IEEE Transactions on Haptics*, vol. 16, no. 2, pp. 118-133, 2023

[6] L. Peternel, T. Petric, E. Oztop, and J. Babič, "Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach," *Autonomous Robots*, vol. 36, pp. 1–14, 01 2014.

[7] M. A. Zamani and E. Oztop, "Simultaneous human-robot adaptation for effective skill transfer," *International Conference on Advanced Robotics (ICAR)*, pp. 78–84, 2015.

[8] M. J. A. Zeestraten, I. Havoutis, and S. Calinon, "Programming by demonstration for shared control with an application in teleoperation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1848–1855, 2018.

[9] Havoutis and S. Calinon, "Learning from demonstration for semi-autonomous teleoperation," *Autonomous Robots*, vol. 43, 03 2019.

[10] L. Peternel and J. Babič, "Humanoid robot posture-control learning in real-time based on human sensorimotor learning ability," *IEEE International Conference on Robotics and Automation*, pp. 5329–5334, 2013.

[11] Y. Michel, R. Rahal, C. Pacchierotti, P. R. Giordano, and D. Lee, "Bilateral teleoperation with adaptive impedance control for contact tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5429–5436, 2021.

[12] L. Peternel, E. Oztop, and J. Babič, "A shared control method for online human-in-the-loop robot learning based on locally weighted regression," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3900–3906, 2016.

[13] W. Si, Y. Guan, and N. Wang, "Adaptive compliant skill learning for contact-rich manipulation with human in the loop," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5834–5841, 2022.

[14] W. Si, T. Yue, Y. Guan, N. Wang, and C. Yang, "A novel robot skill learning framework based on bilateral teleoperation," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pp. 758–763, 2022.

[15] W. Si, N. Wang, and C. Yang, "A review on manipulation skill acquisition through teleoperation-based learning from demonstration," *Cognitive Computation and Systems*, vol. 3, no. 1, pp. 1–16, 2021.

[16] A. Jha and S. S. Chiddarwar, "Robot programming by demonstration using teleoperation through imitation," *Ind. Robot*, vol. 44, pp. 142–154, 2017.

[17] I. Havoutis and S. Calinon, "Supervisory teleoperation with online learning and optimal control," *in 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1534–1540, 2017.

[18] A. Mandlekar, D. Xu, R. Mart'in-Mart'in, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *ArXiv*, vol. abs/2012.06733, 2020.

[19] M. Rigter, B. Lacerda, and N. Hawes, "A framework for learning from demonstration with minimal human effort," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2023–2030, 2020.

[20] C. Zeng, C. Yang, H. Cheng, Y. Li, and S.-L. Dai, "Simultaneously encoding movement and semg-based stiffness for robotic skill learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1244–1252, 2021.

[21] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: object-lifting case study with a humanoid," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pp. 399–404, 2009.

[22] J.-K. Lee and J.-H. Ryu, "Learning robotic rotational manipulation skill from bilateral teleoperation," in *19th International Conference on Ubiquitous Robots (UR)*, pp. 318–325, 2022.

[23] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, 2018.

[24] T. Osa, K. Harada, N. Sugita, and M. Mitsuishi, "Trajectory planning under different initial conditions for surgical task automation by learning from demonstration," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6507–6513, 2014.

[25] K. French, S. Wu, T. Pan, Z. Zhou, and O. C. Jenkins, "Learning behavior trees from demonstration," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7791–7797, 2019.

[26] Konidaris, L. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, 01 2018.

[27] G. Canal, E. Pignat, G. Alenyà, S. Calinon, and C. Torras, "Joining high-level symbolic planning with low-level motion primitives in adaptive hri: Application to dressing assistance," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3273–3278, 2018.

[28] O. Gustavsson, M. Iovino, J. Styrud, and C. Smith, "Combining context awareness and planning to learn behavior trees from demonstration," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1153–1160, 2022.

[29] D.H. Grollman, O.C. Jenkins, "Can We Learn Finite State Machine Robot Controllers from Interactive Demonstration?," *in From Motor Learning to Interaction Learning in Robots. Studies in Computational Intelligence*, vol 264, 2010.

[30] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[31] E. Orendt, M. Riedl, and D. Henrich, "Robust one-shot robot programming by demonstration using entity-based resources,", *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN),* pp. 573–582, 06 2018

[32] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner, and D. Miller, "Interactive hierarchical task learning from a single demonstration," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 205–212, 2015.

[33] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1136–1143, 2021.

[34] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3801–3808, 2015.

[35] Y. S. Liang, D. Pellier, H. Fiorino, and S. Pesty, "Evaluation of a robot programming framework for non-experts using symbolic planning representations," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1121–1126, 2017.

[36] R. Cubek, W. Ertel, and G. Palm, "High-level learning from demonstration with conceptual spaces and subspace clustering," *in 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2592–2597, 2015.

[37] A. Agostini, M. Saveriano, D. Lee, and J. Piater, "Manipulation planning using object centered predicates and hierarchical decomposition of contextual actions," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5629–5636, 2020.

[38] R. Bogue, "Strong prospects for robots in retail", *Industrial Robot*, vol. 46, no. 3, pp. 326-331, 2019.

[39] C. Pezzato, C. H. Corbato, S. Bonhof and M. Wisse, "Active Inference and Behavior Trees for Reactive Action Planning and Execution in

Robotics", *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1050-1069, 2023.

[40] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and ai," *Robotics and Autonomous Systems*, vol. 154, p. 104096, 2022.

[41] O. Biggar, M. Zamani, and I. Shames, "A principled analysis of behavior trees and their generalisations," *ArXiv, vol. abs/2008.11906*, 2020.

[42] F.A. Mussa-Ivaldi, N. Hogan, and E. Bizzi, "Neural, mechanical, and geometric factors subserving arm posture in humans," *The Journal of neuroscience : the official journal of the Society for Neuroscience,* vol. 5(10), pp. 2732–2743, 1985.

[43] L. Peternel, N. Beckers and D. A. Abbink, "Independently Commanding Size, Shape and Orientation of Robot Endpoint Stiffness in Tele-Impedance by Virtual Ellipsoid Interface," *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 99-106, 2021.

[44] C. E. English, D.L. Russell, "Representations of multi-joint stiffness for prosthetic limb design," *Mechanism and Machine Theory*, vol. 43, pp. 297-309, 2008.

[45] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms," *in 2003 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3704-3709, 2003.

[46] G. Franzese, A. Mészáros, L. Peternel, and J. Kober, "ILoSA: Interactive Learning of Stiffness and Attractors," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7778-7785, 2021.

[47] M. Laghi, A. Ajoudani, M.G. Catalano and A. Bicchi, "Unifying bilateral teleoperation and tele-impedance for enhanced user experience," *The International Journal of Robotics Research*, vol. 39, pp. 514-539, 2020.

[48] H. Boessenkool, D. A. Abbink, C. J. M. Heemskerk, F. C. T. van der Helm and J. G. W. Wildenbeest, "A Task-Specific Analysis of the Benefit of Haptic Shared Control During Telemanipulation," *in IEEE Transactions on Haptics*, vol. 6, no. 1, pp. 2-12, 2013.

A

Appendix - ROS Communication Structure Overview

The proposed method for teleoperated teaching of combined high-and low-level skills through a single haptic device is implemented in Robot Operating System (ROS, Noetic). The ROS framework allows for independent design of separate software components (*nodes*) and provides several communication methods for connecting these components. In our implementation, we have leveraged ROS *topics* and *services* to establish connections between the different software modules. Fig. A.1 provides an overview of the communication structure, illustrating the flow of data between Sigma.7, the GUI, the BT (for high-level teaching), and the Cartesian impedance controller (for low-level teaching). Section A.1 elaborates on the main components of the proposed interface and highlights how they are connected. Section A.2 discusses three software components that, while not explicitly shown in Fig. A.1, play important roles in the system's functionality.

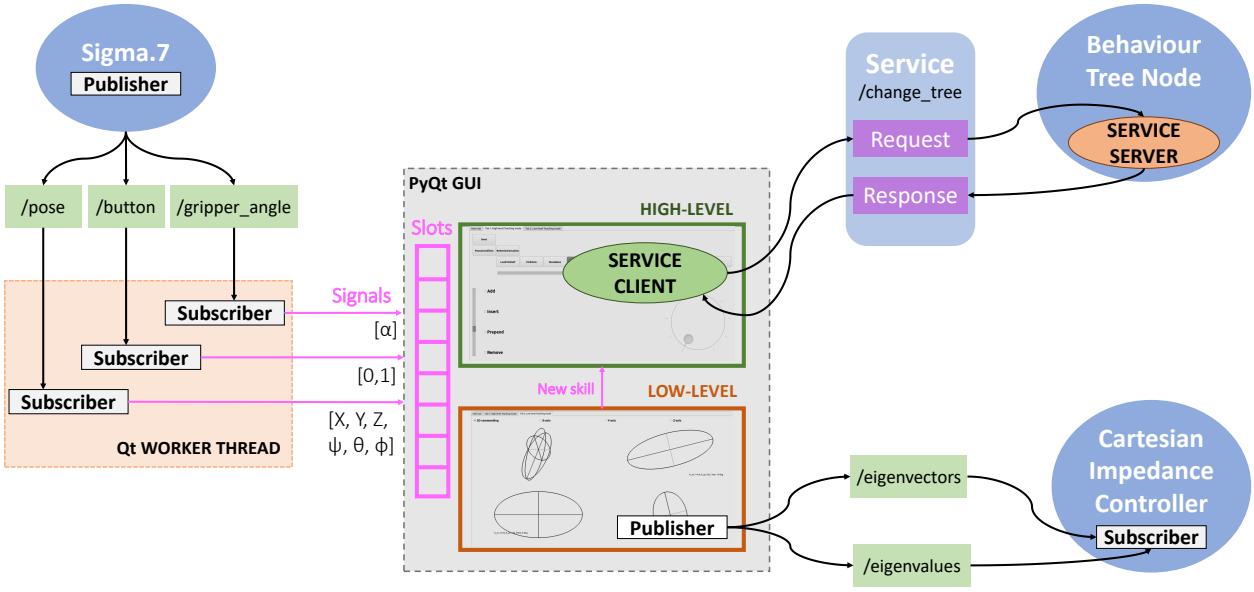


Figure A.1: Overview of the ROS communication structure. Nodes are shown as blue ovals, the orange block denotes the PyQt thread used to handle the incoming data and offload the main GUI, the grey block denotes the GUI, the blue block highlights the *service* used for modifying the BT, *topics* are denoted as green blocks and the pink elements indicate the PyQt *signal/slot* communication. The data flow starts at the Sigma.7 node. The gripper angle signal is denoted with α, the binary state signal of the emulated button click with [0,1], and the 6 DoF haptic device signals are denoted with X, Y, Z, ψ, θ, ϕ. Note: for brevity, we represent the two subscribers within the "Cartesian Impedance Controller" nodes using a single block. However, both topics of course have separate subscribers.

A.1. Main Components

The communication structure is composed of five main connections:

- **Sigma.7 → Qt Worker Thread**: The Sigma.7 haptic device publishes onto several topics. We utilize three streams of data. The `/pose` topic receives the 6 DoF pose (position [m] and orientation [rad]) of the haptic device endpoint that is controlled by the human operator. The `/button` topic receives the binary state of the emulated button click action, executed with the finger-operated button. When the state goes from 0 to 1 this indicates a button click. The `/gripper_angle` topic receives the angular position in [rad] of the finger-operated gripper of Sigma.7. We have implemented a PyQt worker thread (orange block in Fig. A.1) dedicated to listening to these incoming data streams. The use of threads for subscribing to the topics was necessary to prevent freezing of the PyQt GUI.

- **Qt Worker Thread → PyQt GUI**: The subscribers in the Worker Thread are connected to callback functions. When data arrives at these callback functions, they trigger the emission of PyQt signals. A signal essentially is a message sent by one part of the program (the Worker Thread) to another (the PyQt GUI). The GUI has specific predefined slot functions for each possible signal. These slot functions interpret the data, update the GUI and trigger specific actions in the GUI based on the received information. Repurposing of the haptic device was achieved by making effective use of the signal/slot mechanism, as it allowed for using the data for specific purposes while efficiently "ignoring" data that is not currently needed. In the high-level GUI, the slot functions were used to control the sliders, button clicks and dial. In the low-level GUI, the slot functions were used to scale the 6 DoF haptic device endpoint data to match the screen width and height. The scaled translational inputs were used to set the principal axes lengths of the ellipsoid render. The scaled rotational inputs were used to define the rotation of the ellipsoid. This way, the user is able to manipulate the virtual ellipsoid in 6 DoF. Furthermore, the button click slot function was used to select the axis to control. The signal/slot communication mechanism is indicated in pink in Fig. A.1.

- **High-level GUI → Behaviour Tree Node**: When high-level teaching is selected, the user operates the high-level GUI with the haptic device. The interaction with the GUI follows a three-step process: 1) selection of a BT node[1], 2) selection of an operation to perform on the node, 3) if the insert operation is selected, specifying the desired index. The GUI guides the user through these steps, ensuring that only the relevant options are enabled and responsive at any given moment. For example, when selecting a node, the options to choose an operation and specify an index are temporarily greyed out and unresponsive to haptic device inputs until the user has completed the previous step. To establish communication between the high-level GUI and the BT for implementing the requested changes, we designed a custom service (`/change_tree`) and a corresponding message type. The high-level GUI is the service client and the "Behaviour Tree" node is the server. A request message is sent from the client to the server, which contains 1) the name of the node, 2) the operation to be executed on the the node, 3) if applicable, the index at which a node must be inserted. Upon receiving this request message, the server processes the user's command and modifies the BT. It then sends a response message back to the client, indicating whether the requested change was successfully implemented.

- **Low-level GUI → Cartesian Impedance Controller**: When low-level teaching is selected, the service `/controller_manager/switch_controller` (this service is not shown in Fig. A.1) is used to start the Cartesian impedance controller. The user operates the low-level GUI with the haptic device. The user can change size, shape and orientation of the endpoint stiffness ellipsoid of the robot by moving the haptic device in 6 DoF. The ellipsoid graphics are rendered by implementing a `QOpenGLWidget` in the low-level PyQt GUI. To continuously command endpoint stiffness, we chose topics as the communication type. The low-level GUI publishes the commanded principal axes lengths of the ellipsoid (i.e., the eigenvalues of the stiffness matrix), and the commanded orientation with respect to the robot base frame (i.e., the eigenvectors) onto the respective topics, `/eigenvalues` and `/eigenvectors`. The "Cartesian Impedance Controller" node is subscribed to these topics and uses the data to construct the endpoint stiffness matrix.

[1] Note the difference between ROS nodes and Behaviour Tree nodes

- **Low-level GUI → High-level GUI**: As visualized by the pink arrow between both windows (“New skill”), there is also a signal/slot connection present to allow for use of the newly demonstrated low-level skill in the BT. Once the low-level teaching has finished, a signal is emitted from the low-level window to the high-level window to request for addition of a new BT node to the skill library. Furthermore, upon receiving this signal, the high-level window uses the service `/controller_manager/switch_controller` to stop the Cartesian impedance controller and start the position controller used by `MoveIt`.

A.2. Additional Functionalities

A.2.1. Window switching

The START window of the GUI, used to select a teaching mode, is not visualized in Fig. A.1. However, this GUI part does have one important role we need to address. The high- and low-level GUI windows use the same incoming signals and therefore both have slot functions that simultaneously respond to a signal. An important design challenge was ensuring that only the actively selected teaching mode responds to these signals while preventing the non-selected mode from doing so. To solve this problem, we designed the GUI as a `QTabWidget`. When a user selects either the high-level or low-level teaching mode by switching tabs, the `QTabWidget` registers this change. The START window of the GUI continuously stores the index of the currently opened tab (i.e., the high- or low-level window) and publishes this value onto a designated topic. The Qt Worker Thread has a separate subscriber for listening to these GUI tab changes. The index of the currently opened tab is sent as a signal to the high-and low-level teaching modes. This then ensures that only the slot functions associated with the active tab remain responsive to incoming haptic device-related signals. This solution is illustrated in Fig. A.2.

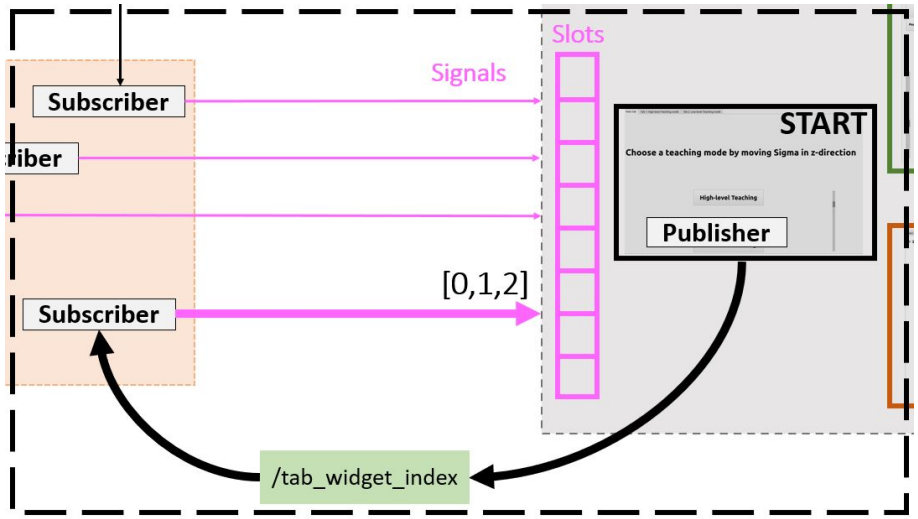


Figure A.2: The role of the START window of the GUI. We implemented a *topic*-based solution implemented to avoid interference between slot functions for both teaching modes. The main window has index 0, the high-level teaching window has index 1 and the low-level window has index 2. This illustration extends Fig. A.1, in which the START window was left out for brevity.

A.2.2. Kinematic Trajectory for Low-level Teaching

Once the human operator starts recording of the stiffness by using the emulated button click action (executed with the finger-operated gripper of the haptic device), a predefined kinematic trajectory is sent to the Cartesian impedance controller. This is visualized in Fig. A.3. To offload the GUI, we created a separate thread to handle the motion trajectory playback. The motion trajectories required to reach the shelf, from both the perpendicular position (Section III.C) and the position at a 45-degree angle relative to the shelf (Section III.D) were established by trial-and-error. The trajectory is composed of several waypoints that finally position the end-effector of the robot in the right location to drop off the product. The thread publishes the waypoints that create the trajectory to the `/equilibrium_pose` topic. This

equilibrium pose is the desired or reference position, i.e. x_r in Eq. 5 in Section II. The "Cartesian Impedance Controller" node continuously publishes the current or actual end-effector position, i.e. x_a, and orientation to `/cartesian_pose`.

A.2.3. RViz Ellipsoid Visualization

Fig. A.3 shows the ROS nodes and topics used to provide the human operator with an RViz visualization of the endpoint ellipsoid he/she is commanding. The visualization was based on a simple example provided in [1]. The "Ellipsoid visualization" node is both a subscriber and a publisher. It listens to the data being streamed onto `/eigenvalues` and `/eigenvectors`, to get the size, shape and orientation of the ellipsoid. Furthermore, it subscribes to the `/cartesian_pose` topic, so the ellipsoid can correctly be positioned at the end-effector of the robot. It publishes a `Marker()` message to the `/visualization_marker` topic, which is used by RViz to create the ellipsoid.

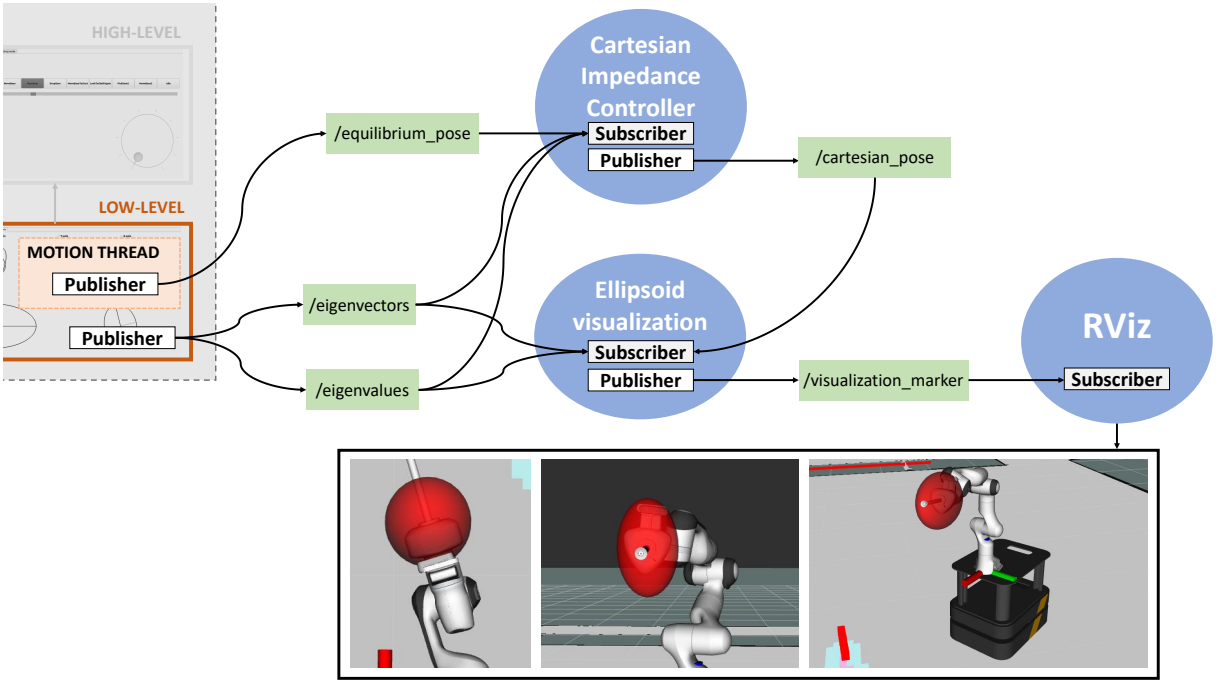


Figure A.3: ROS nodes and topics used to 1) send the pre-configured kinematic trajectory to the Cartesian impedance controller, and 2) to create a visualization of the endpoint ellipsoid in RViz. The RViz screenshots show three different views of the endpoint ellipsoid. From left to right: X-Y plane, Y-Z plane and 3D. Note: for brevity, we represent the multiple subscribers within the "Cartesian Impedance Controller" and "Ellipsoid Visualization" nodes using a single block. However, each topic of course has separate subscribers. This overview extends Fig. A.1.

B

Appendix - Predefined Behaviour Tree

We employed Behaviour Trees (BTs) as the model to represent the high-level decision-making for a pick-and-place task in a supermarket environment. In the high-level teaching mode, as explained in Sections II.B, III.B and III.E, the human operator uses the haptic device to remotely modify the action sequence encoded by the predefined BT. Section B.1 provides an explanation of the fundamental principles underlying BTs. In Section B.2, we examine the BT constructed for this research and we provide recommendations for future work.

B.1. Preliminaries

BTs represent the state transition logic through an hierarchical tree structure [2]. BTs map states to actions. The tree-structured diagrams represent a policy, i.e. they specify the order and relationships between the available actions, and take into account the conditions under which those actions must be executed. There exist two types of nodes used to construct a BT: control flow nodes (*Sequence, Fallback, Parallel, Decorator*) and execution nodes (*Action, Condition*). Action nodes are referred to as leafs as they have no children. They reflect the physical (i.e., low-level) skills of the robotic system. The root node of a BT starts the execution by sending so-called *ticks* at a certain frequency to its children. The tick propagates through the tree. A node is executed if and only if it receives ticks [3]. The execution nodes return either *Running, Success* or *Failure* to the parent node. These outputs of the ticked node determine which action is taken next. The ticking process guides the exploration of the BT.

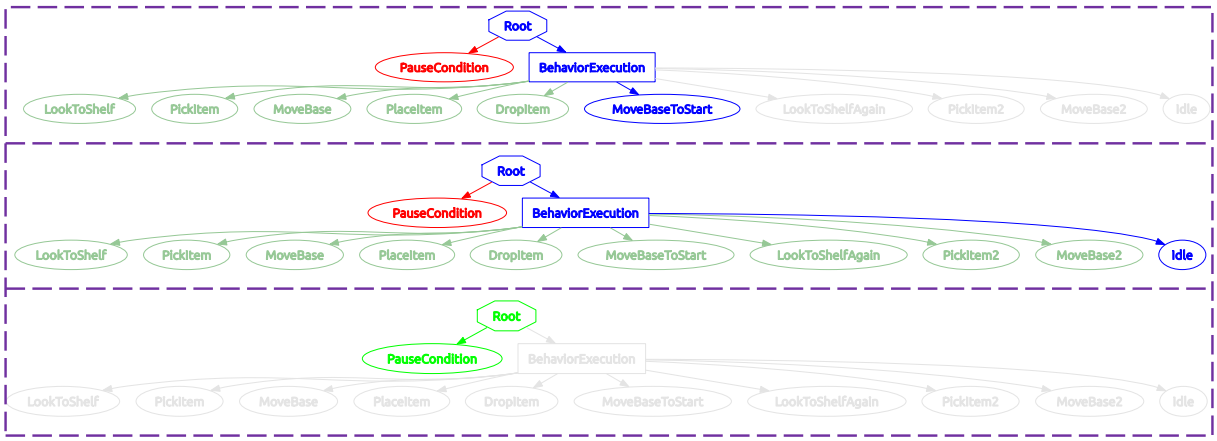


Figure B.1: The predefined BT used for this research at three different modes. The colours indicate the state of the nodes: blue means “running”, green is “success”, red is “failure” and grey denotes “unvisited”. Top: the BT children are being executed in sequential order. Middle: all children are successfully executed and BT remains in idle mode. Bottom: the “PauseCondition” node returns success, and because the “Root” node is a *Fallback* node, this prevents “BehaviorExecution” from being ticked.

B.2. Pick-and-Place BT

We constructed the predefined BT using `py_trees_ros`, which occurs in the “Behaviour Tree Node” shown in Fig. A.1 in Appendix A. Fig. B.1 shows the full BT in three different states. This visualization of the BT was obtained with `rqt_py_trees`.

The “Root” node is a fallback node, and ticks its first child: “PauseCondition”. This is a condition node, that checks if the tree execution must be paused (for example when the Low-level teaching mode is started). If the condition returns False, the root ticks the next child: “BehaviorExecution”, which is a sequence node. A sequence node succeeds only if all children return success. All children of this node are action nodes and executed in the following order: “LooktoShelf”, “PickItem”, “MoveBase”, “PlaceItem”, “DropItem”, “MoveBaseToStart”, “LooktoShelfAgain”, “PickItem2”, “MoveBase2”, “Idle”. Execution of these manually coded low-level skills occurs through the ROS `actionlib` Action Server/Client interaction mechanism. The nodes in the BT are the clients that request goals, which are executed by their corresponding server. The action servers use *MoveIt* for motion planning and are launched with a dedicated `roslaunch` file.

B.2.1. Future Work

As follows from Fig. B.1, the predefined BT encodes the behaviour for two different pick-and-place sequences: once the first product is placed in the shelf, the robot returns to the initial base location to start a second pick-and-place sequence. This was necessary, because the `py_trees_ros` package for ROS1 only allows for sending pre-configured goals to the `actionlib` action server. However, `py_trees_ros` for ROS2 does have the functionality to send new goals to the server at runtime through the blackboard. To increase modularity of the BT, migrating all software parts to ROS2 is an essential recommendation for future work.

Furthermore, the current design of the BT follows an “explicit design”. To increase the robustness and reactivity of the BT an “implicit” sequence is necessary. To achieve this, the the sequence “BehaviorExecution” node must be replaced by a fallback node, the order of the low-level actions must be reversed and the appropriate preconditions must be added, following a postcondition-precondition-action pattern (PPA) [3] [4].

C

Appendix - Dynamic Movement Primitives

Ultimately, the proposed shared control interface can facilitate the machine learning aspect of LfD on both a high- and low-level. We took initial steps of using the interface for low-level skill learning, by using Dynamic Movement Primitives (DMPs) to encode the stiffness trajectories. Section C.1 provides an overview of the DMP framework. In Section C.2, we present preliminary results that show our progress in using DMPs to reproduce a skill that was demonstrated with our proposed interface.

C.1. Preliminaries

C.1.1. Classical Point-to-Point DMP Formulation

The main idea of DMPs, as introduced in [5], is to describe movements as a set of differential equations that ensures some desired behaviour. DMPs are defined for discrete and rhythmic movements. Given that we are only interested in discrete point-to-point motions, we focus on the point attractor formulation (discrete DMPs) [6]. A DMP for a single DoF trajectory y of a desired discrete movement is defined by the following set of nonlinear differential equations [7]

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x) \tag{C.1}$$

$$\tau \dot{y} = z \tag{C.2}$$

$$\tau \dot{x} = -\alpha_x x \tag{C.3}$$

where g is the goal configuration, z is the scaled velocity and x is the phase variable, which serves as a replacement of time, α_z and β_z are positive constants that define the 2nd-order system, and $f(x)$ is a nonlinear forcing term which enables a smooth trajectory from the initial position y_0 to the final configuration g. By choosing time constant $\tau > 0$ and $\beta_z = \alpha_z/4$ (critical damping), and $\alpha_x > 0$, convergence of the system to the unique attractor point at $y = g$ and $z = 0$ is ensured. The forcing term $f(x)$ can be learned from demonstrations and is defined as

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x \tag{C.4}$$

with N nonlinear Radial Basis Functions (RBFs),

$$\Psi_i(x) = \exp\left(-h_i(x - c_i)^2\right) \tag{C.5}$$

where c_i and h_i are the centers and the widths of Gaussians distributed along the movement phase. For each DoF, the weights w_i must be adjusted to achieve the desired movement to the goal. Learning these weights, i.e. learning $f(x)$, is typically achieved with Locally Weighted Regression (LWR) [8]. Every DoF has its own *transformation system* (Eq. C.1-C.2), but with a common *canonical system* (Eq. C.3) to synchronize them. DMPs describe the dynamical system as a damped spring model, that pulls a DoF to a goal state along a learned trajectory.

C.1.2. Using DMPs to encode SPD matrices

Our goal is to learn stiffness trajectories using DMPs. The classical formulation of discrete DMPs as presented in Eq. C.1-C.3, is limited to data in Euclidean space, i.e. when the evolution of movements in different DoF is independent [8]. Therefore, DMPs cannot directly be applied for learning skills that are subject to specific geometry constraints, such as the SPD stiffness matrix. In order to use DMPs to encode stiffness skills, [9] extended the DMP framework to allow for encoding SPD matrix trajectories. Their proposed method for "Geometry-aware" DMPs is based on exploiting Riemannian metrics to reformulate DMPs, such that the set of SPD matrices forms a Riemannian manifold. Using the properties of Riemannian manifolds, they prepare the demonstrated SPD matrices for DMP formulation. In-depth discussion of these steps is outside the scope of this Appendix.

The resulting geometry-aware DMP formulation is as follows. By defining $\mathbf{X} \in \boldsymbol{S}_{++}^{m}$ as an arbitrary SPD matrix, and Ξ as the set of SPD matrices recorded in one demonstration, where $\boldsymbol{S}_{++}^{m}$ defines the set of $m \times m$ SPD matrices, they rewrote Eq. C.1 and C.2 as:

$$\tau\dot{\boldsymbol{\sigma}} = \alpha_z \left(\beta_z vec\left(\mathbb{B}_{\mathbf{X}_l \to \mathbf{X}_1}\left(\mathrm{Log}_{\mathbf{X}_l}\left(\mathbf{X}_g\right)\right)\right) - \boldsymbol{\sigma}\right) + \boldsymbol{F}(x) \tag{C.6}$$

$$\tau\dot{\boldsymbol{\xi}} = \boldsymbol{\sigma} \tag{C.7}$$

where $\boldsymbol{\sigma} = vec(\boldsymbol{\Sigma})$ is the Mandel representation of $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}$ is the time derivative of Ξ and $\boldsymbol{\xi}$ is the vectorization of Ξ. $\mathbf{X}_g$ is the goal SPD matrix. $\boldsymbol{F}(x)$ is the recalculated forcing term. For an extensive explanation of all terms, we refer to [9].

C.2. Reproduction of the demonstrated stiffness trajectory

We applied the geometry-aware DMP framework as introduced by [9] to a dataset of SPD matrices recorded with the low-level teaching mode of our proposed interface. We used the code that was made available in [8]. The data was recorded during the experiment described in Section III.D: placing the product at the designated location at a 45 degree angle with respect to the shelf. During demonstration, the stiffness matrices and the Cartesian end-effector positions of the robot were recorded using the `rosbag` package.

C.2.1. Data Pre-processing

As the product placement required a variable stiffness trajectory in the X and Y directions, the recorded 3×3 stiffness matrices $\boldsymbol{K}$ were simplified into 2×2 matrices (excluding the Z direction dependent elements) to remove any unnecessary complexity of the dataset.

The GUI dictates the update rate of the stiffness matrix, while the Cartesian impedance controller node determines the update rate of the Cartesian end-effector position. Because the update rate of the GUI is considerably lower, the number of recorded stiffness matrices did not match the number of recorded poses, necessitating a pre-processing step. This step involved comparing the recorded timestamps to filter out redundant end-effector poses.

C.2.2. Results

Fig. C.1 shows the geometry-aware DMP reproduction of the variable stiffness trajectory that was demonstrated with our virtual ellipsoid interface. During the demonstrated product placement, the human operator increases stiffness in Y direction and rotates the ellipse to ensure precise alignment, see top graph in Fig. C.1. This demonstration is then encoded with the geometry-aware DMPs using Eq. C.6-C.7 to reproduce the green ellipses in the bottom graph of Fig. C.1. There is match between the demonstration and the reproduced skill: stiffness in Y is increased and the ellipses is rotated 45 degrees. However, it is clear that the learned skill does not yet perfectly match the demonstration. This could be due to the following reasons.

First, it is important to note that we have not conducted any parameter tuning of the constants in Eq. C.6-C.7 yet. We used the default settings from the open source implementation [8] for the control gains $\alpha_x, \beta_z, \alpha_x$ and the number of Gaussian basis functions in the forcing term N. While these settings provide a starting point, they should be fine-tuned to increase performance.

Second, the mismatch could be attributed to the low refresh and update rate of the GUI. To generate smooth trajectories using DMPs, smooth training data is required [5]. A smooth trajectory is characterized by minimal jerk, which implies that the time derivative of acceleration must be minimized. Hu-

man movements naturally exhibit this characteristic [7], referred to as minimum-jerk trajectories. Even though the human operator smoothly modulates the size and shape of the ellipsoid, the low update rate of the GUI leads to an insufficient number of data points being recorded. This leads to abrupt changes in the recorded stiffness matrices between consecutive time steps. Consequently, this compromises the minimum-jerk aspect of the demonstrated trajectory. This lack of data points results in the removal of the inherent smoothness that is associated with the demonstrated skill by the human operator. The consequence of these sharp discontinuities in the recorded data is evident in Fig. C.1, where the DMP cannot handle such abrupt changes effectively. To address this challenge and enhance the compatibility of the recorded data with the DMP framework, future work should explore interpolation techniques that can mitigate the impact of abrupt changes caused by the low GUI update rate and restore the smoothness of the training data.

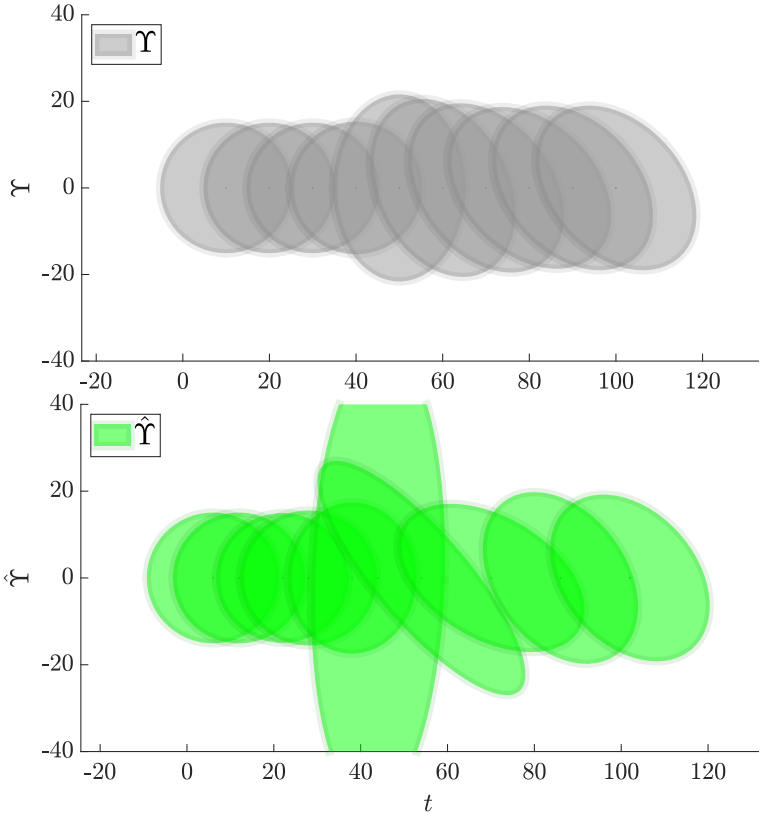


Figure C.1: Learning and reproduction of a variable stiffness trajectory using SPD DMPs [9] for product placement at a 45 degree angle relative to the shelf. Top: demonstrated stiffness ellipses in the X-Y plane in $T = 102$ s. This training data is used to learn the weights of $N = 40$ Gaussian basis functions. Bottom: the reproduced stiffness ellipses over time. For clarity, not all ellipses are plotted. Results are obtained with the open source implementation available at `https://gitlab.com/dmp-codes-collection` [8].

Bibliography

[1] T. Coleman, G. Franzese, and P. Borja, “Damping design for robot manipulators,” in *Human-Friendly Robotics 2022: HFR: 15th International Workshop on Human-Friendly Robotics*, Springer, 2023, pp. 74–89.

[2] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, “A survey of behavior trees in robotics and ai,” *Robotics and Autonomous Systems*, vol. 154, p. 104 096, 2022, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2022.104096. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889022000513.

[3] M. Colledanchise and P. Ogren, *Behavior Trees in Robotics and AI: An Introduction*. Jul. 2018, ISBN: 9781138593732. DOI: 10.1201/9780429489105.

[4] E. Dortmans and T. Punter, “Behavior trees for smart robots practical guidelines for robot software development,” *Journal of Robotics*, vol. 2022, pp. 1–9, Sep. 2022. DOI: 10.1155/2022/3314084.

[5] A. Ijspeert, J. Nakanishi, and S. Schaal, “Trajectory formation for imitation with nonlinear dynamical systems,” vol. 2, Feb. 2001, 752–757 vol.2, ISBN: 0-7803-6612-3. DOI: 10.1109/IROS.2001.976259.

[6] A. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” vol. 2, Feb. 2002, pp. 1398–1403, ISBN: 0-7803-7272-7. DOI: 10.1109/ROBOT.2002.1014739.

[7] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013. DOI: 10.1162/NECO_a_00393.

[8] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, *Dynamic movement primitives in robotics: A tutorial survey*, 2021. arXiv: 2102.03861 [cs.RO].

[9] F. J. Abu-Dakka and V. Kyrki, “Geometry-aware dynamic movement primitives,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4421–4426. DOI: 10.1109/ICRA40945.2020.9196952.