Robotic Skill Mutation when Propagating a Physical Collaborative Task from Robot-to-Robot.

Rosa E.S. Maessen

Master of Science Thesis

TU Delft Delft University of Technology

Department of Cognitive Robotic, TU Delft

Robotic Skill Mutation when Propagating a Physical Collaborative Task from Robot-to-Robot.

by

Rosa E.S. Maessen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on December 16th 2022.

Student number:	4564200	
Master program:	Robotics	
Thesis committee:	Dr. L. Peternel	TU Delft, supervisor, chair
	Prof. Dr. M. Wisse	TU Delft, member
	Dr. A. Seth	TU Delft, external member
	Dr. J.M. Prendergast	TU Delft, added member

TU Delft

PREFACE

This thesis, named "Robotic Skill Mutation when Propagating a Physical Collaborative Task from Robot-to-Robot", is the culmination of my research for my Master's in Robotics at the Delft University of Technology. Over the course of the past year, I have had the privilege of exploring the potential of skill mutation while conducting a collaborative sawing task. In the final months of this research, I was able to take advantage of the KUKA LBR robots available at the university and apply my method in a real-world setting. I had limited experience in programming robots for real-world applications before, so working with these robots was incredibly valuable to my learning process.

The results of my research show that skill mutation does occur during the propagation of collaborative tasks. Additionally, I gained insight into why this mutation occurs and recognised its potential benefits and risks. This thesis provides a thorough overview of my research and results and a detailed discussion of the implications of my findings. I hope this thesis will be useful for future researchers exploring skill mutation in a collaborative setting.

I am incredibly grateful to all of those who have contributed to making this research successful. I want to thank Luka Peternel, my supervisor, for his consistent guidance and feedback during our weekly meetings. Micah Prendergast deserves special mention for his technical support with the KUKA robots and for his willingness to discuss the project during the final months of the thesis. Furthermore, I am thankful for the invaluable assistance from the PhD candidates of the Cognitive Robotics department, who assisted me with the KUKA robots and provided insight into potential solutions. Lastly, I extend my sincerest gratitude to my friends who have made graduating an enjoyable experience. They have provided me with emotional support and constructive feedback, especially on my grammar, which has been invaluable in the completion of my thesis. Without their support, this journey would not have been possible.

Rosa E.S. Maessen
Delft, December 2022

CONTENTS

Robotic Skill Mutation when Propagating a Physical Collaborative Task from Robot-to-Robot.

Rosa E.S. Maessen[1]
Supervised by: Luka Peternel[1]

***Abstract*—In this research, we examined the occurrence of skill mutation when propagating a collaborative sawing task from robot-to-robot. We conducted this research, to gain insight into this mutation to understand the generation of potentially beneficial or dangerous skills. Thirty propagation steps in simulation were conducted per experiment, each consisting of one expert robot teaching a novice (learner) robot. To explore what influences mutation, different external factors were changed, such as the maximum stiffness of the robots, the base position of the robots, the friction coefficient of the object and saw, and the period span of one sawing movement. The robots were controlled using a hybrid force/impedance controller, with the impedance part responsible for the sawing movement. The goal of the skill propagation was to teach the novice robot the impedance controller inputs (desired trajectory and stiffness), which is implemented through a three-staged learning process. In stage one, the desired trajectory was learned by encoding the measured trajectory using Dynamic Movement Primitive (DMP) and Locally Weighted Regression (LWR), in stage two the stiffness was learned by encoding the computed stiffness, and in stage three the novice robot became an expert, able to collaboratively execute the task. The results showed that the skill varied over the different propagation step, therefore proven the existence of skill mutation. It was found that the biggest mutations were caused by a phase lag, overshoot on the desired trajectory, differences in joint states, and reaching torque limits. It was also found that environmental boundaries limited the mutations. By comparing the results of the different propagation steps of both different and the same conditions, it was that the mutations were not reproducible. This is a result of not being able to fix all external factors. We also identified the benefits (skill useful for different settings or different tasks, and energy efficiency) and dangers/drawbacks (high forces and skill becoming useless for initial task) of the mutation.**

***Index Terms*—Robotic skill mutation, skill propagation, collaborative task, robot-to-robot learning, periodic Dynamic Movement Primitive (DMP) and Locally Weighted Regression (LWR).**

I. Introduction

When many robots need to be programmed at once, the conventional method of programming each robot individually might not be practical. This is particularly true when, for example, the robots are of different types or when they are subject to different environmental constraints. A solution could be to teach one robot a task, after which that robot can propagate its skill to the other robots. This concept of skill propagation is illustrated in Fig. 1. While each of the robots could learn a nearly identical behaviour from their teacher, skill propagation could also result in different behaviour.

[1]Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands.

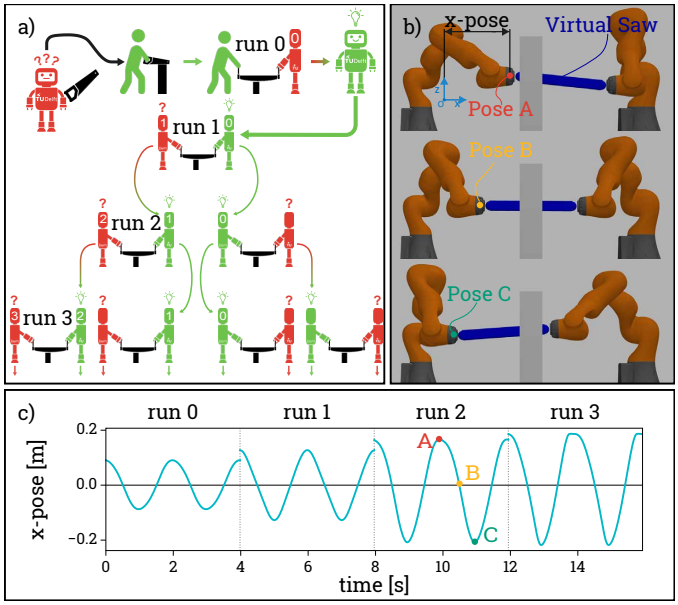


Fig. 1. Concept of illustration of robot skill propagation. a) Shows the process of skill propagation using a sawing task. A human demonstrates the task to a novice robot (robot 0, illustrated in red), who then collaboratively executes the task with the human, which allows the robot to learn the task and become an expert (illustrated in green). The robot is then able to propagate its learned skill to other robots. b) Shows a sequence of images which indicate different moments of the sawing movement. c) Shows a graphical representation of the movement of the robot along the x-axis during 4 s of four different runs (identified in image a). Points A, B and C refer back to the sequence of images in b).

For example, the robot could learn to move to a desired position using a different trajectory than what was taught by their teacher. These differences would, ideally, improve the performance of the task. Alternatively, these differences could also lead to unsafe behaviour or the skill becoming useless for the original task. For the latter, it is possible that the skill would be useful for a different task. Mutation due to skill propagation is the topic of this research.

Robot skill acquisition has been researched extensively in literature, and many possibilities can be found. Up to the 1980s, manually programming the skill of the robot was the most frequently used approach. Here, a code is created, usually consisting of hundreds or even thousands of parts, which combined describe the behaviour of the robot. Especially in industrial settings where robots are required to perform a repetitive task, this method is still frequently used [1]. However, the demand for humans to work alongside robots has increased in recent years. Due to safety issues, the usage of manually

programmed robots has become an issue, as they are unable to deal with unpredictable environments [2, 3]. In addition, manually coding robots is not intuitive and time-consuming [4]. Therefore, recent studies have focused on the possibility of robotic learning, which allows the robot to learn and adjust its skill by interacting with the environment and/or other agents. Robotic learning also allows for the generalizability of the skill, which means that the skill, learned in a specific setting, could be exploited in other settings [5].

One widely studied approach, which allows for learning from the environment, is Reinforcement Learning (RL). RL uses a trial-and-error approach, where the environment is explored in order to perform the desired task as accurately as possible [6–9]. The learning of the robot is guided by a human-defined cost or reward function that motivates the robot to improve the task performance during the different iteration steps. An example is the dart game, where a robot should learn to hit a target on a board [10]. The reward of the robot could then be dependent on the distance between the position of the dart after throwing, and the target position. Reward functions can also be discrete, where, for example, a positive reward is given if a goal is reached and/or a negative reward is given if the robot is not able to complete the task [11]. In [12], a more complex reward function is used for a picking task. In addition to the difference between the desired and actual position, the approach also considered the amount of reactive control needed to reduce the occurrence of irreversible events like slipping. RL has the advantage of searching for a solution without the need for human involvement, which allows for the exploring of many possible solutions for a task. However, this also has the downside that, especially during the initial exploratory actions, unsafe behaviour both for the robot and its environment can occur [11]. In addition, often many iterations are needed until a solution is found, which makes this approach data-extensive and time-consuming and thus undesired for a real-world implementation. An alternative approach that overcomes these limitations is Learning from (human) Demonstration (LfD), which comes at the expense of human involvement.

LfD, also known as Programming by Demonstration (PbD), Imitation Learning (IL) and apprenticeship learning, uses (human) demonstrations to train an agent [3, 13–15]. The demonstration could be exerted by kinaesthetic teaching (guiding the robot) [16–20], teleoperation (controlling the robot using an external device) [21–23], or passive observation (observing the execution of the expert and perform the task based on this data) [24, 25]. A big advantage of this approach is that the human can demonstrate safe skills directly. However, as the robot learns directly from an expert, the skill is limited by the ability of the expert to perform a task or provide a good demonstration [14]. To tackle this issue, LfD can be combined with an exploration-based method like RL [26, 27], which can improve upon initial non-optimal human demonstrations. However, these previous approaches were limited to single-agent tasks and did not account for direct collaborative tasks. In [28], a method was presented to teach the skill through online collaboration, where a combination of demonstration and optimisation is employed. The results of the experiments conducted in this research showed variability in the behaviour of the learning agent compared to the teacher. Therefore, one can use this method to avoid the limitation posed by the ability of the expert to perform the desired task.

We define this deviation of the skill between the novice (learner) and the expert as skill mutation. In [28], the goal was to learn an impedance control, consisting of a desired trajectory and corresponding stiffness, in order to execute a collaborative task involving physical interaction between two agents. It could be observed that mutation occurred both for the learned trajectory, describing the desired position of the robot, and the stiffness of the robots, describing the influence of the specific robot on the task. While it is visible that new behaviour has emerged, this study has not explored the underlying reasoning for these mutations. Therefore, there is a knowledge gap in why skill mutation occurs during robot-to-robot skill transfer. It is important to have insight into these mutations to understand the generation of potentially beneficial or dangerous skills.

To address this critical gap, we performed a study to explore the mutation of skill when learning a collaborative task from an expert online during task execution. The benefit of online learning is the ability of the agent to adjust its skill based on the interaction with the other agent, and the environment [29, 30]. Especially for a collaborative task, this is important, as the execution only partially depends on the skill of one agent. Instead, it depends on the interaction between both agents. The following research question guided the investigation of the skill mutation:

> *When a skill mutation results from doing robot-to-robot learning, in what way does the skill mutate and what causes this mutation?*

To answer the research question, we investigated three subparts. Firstly, we examined whether mutation occurs and, if so, how it mutates. Secondly, we tested multiple external factors to observe if, and if so, how they influence the mutation. The factors discussed in Section III-D are as follows: the maximum stiffness of an agent, the base position of the robot, the period describing the time that one sequence of the movement requires, and the friction force acting in the opposite direction of the movement. Lastly, we investigated the repeatability of the mutation when using the same conditions. The experiments conducted to investigate each of these subjects are described in Section III-A.

To better understand how the skill mutations occur, we investigated the mutation of a robotic skill by performing simulations and experiments on two KUKA LBR iiwa robots performing and learning a collaborative sawing task. During the initial propagation of the skill, one robot was an expert while the other was a novice without skill. To teach the novice robot the skill, we extended an online learning approach based on [28]. After learning, the novice was assumed to have become an expert, which was followed by this new expert propagating its learned skill among other novice robots in a similar manner.

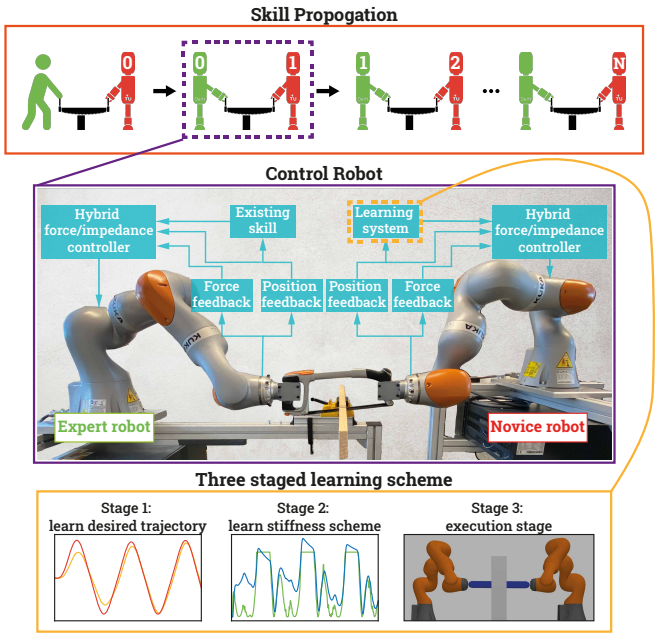


Fig. 2. Workflow of the skill propagation.The robot propagate its skill to another by executing a collaborative task. The expert (teacher) robot is here indicated in green, and the novice (learner) in red. To control the robots, a hybrid force/impedance controller is used. The impedance part of this controller makes use of either an existing skill (for the expert) or a three staged learning scheme (for the novice). The three staged scheme consist of a stage in which the desired trajectory is learned, one where the stiffness is learned, and one where the learned skill is used to execute the task.

II. Methodology

In this research, a collaborative sawing task is used to investigate the mutation of robotics skills during skill propagation. The robot is controlled using a hybrid force/impedance controller, where the force control is responsible for maintaining contact with the environment, and the impedance control for the sawing movement (Fig. 2). During the collaborative execution of the sawing task, one novice agent will become an expert. Therefore, it can propagate its learned skill to the next robot, and so on. This skill propagation process is expanded in Section II-B. To learn the sawing task, a three-staged learning scheme is used (Section II-C). The goal of this scheme is to learn the different elements of the impedance controller, which are encoded using DMPs with LWR (Section II-D).

A. Task and Robot Control

This research uses a collaborative task, which should allow for more variability in the learned skill. This variability is a result of not precisely dictating the movement of the robot during learning, which is the case for the previously discussed demonstration methods (kinaesthetic teaching/teleoperation/passive observation). In addition, the performance of the task is not just affected by one agent, which makes the ability of the agents to interact with one another a key factor of the task. For this purpose, we use a collaborative sawing task. This task has the advantage that the skill consists of a periodic movement, which allows for the evolution of the mutation to be clearly visible. In addition, it incorporates the interesting feature of leader/follower behaviour that is periodically exchanged during the execution. When executing the collaborative sawing task, it is desired to have a high stiffness when pulling (leader) and a low stiffness (high compliance) when pushing (follower) in order to oppose each other in different stages [31].

The collaborative robots are controlled using a hybrid force/impedance controller, which allows for the control of the robot to be separated into two subspaces [32]. The force controller ensured that the saw maintained contact with the environment by exerting a specific amount of force. The impedance controller was used to control the movement while dealing with the uncertainties of the environment [31].

$$\boldsymbol{F}_{control} = \boldsymbol{F}_{for} + \boldsymbol{F}_{imp}, \tag{1}$$

where $\boldsymbol{F}_{control}$ is the control force resulting from the hybrid controller, $\boldsymbol{F}_{for}$ the force controller, and $\boldsymbol{F}_{imp}$ the impedance controller.

To maintain a desired force, a PI controller was implemented. The PI is preferred over the PID controller when the measured signal is noisy, which is the case of the force readings as the saw interacts with the environment[31]. Since digital controllers are implemented with discrete sampling periods, a discrete form of the PI controller has been used

$$\boldsymbol{F}_{for} = \boldsymbol{K}_P^F \boldsymbol{e}_F + \boldsymbol{K}_I^F \sum_{i=0}^{n_t} \boldsymbol{e}_{F,i}(t)\Delta t, \tag{2}$$

$$\boldsymbol{e}_F = \boldsymbol{F}_d - \boldsymbol{F}_a, \tag{3}$$

where $\boldsymbol{e}_F$ describes the difference between the measured force $\boldsymbol{F}_a$ acting on the end-effector of the robot and the desired force $\boldsymbol{F}_d = -5\,\mathrm{N}$. The PI controller tuning values are set by the diagonal matrices $\boldsymbol{K}_P^F$ and $\boldsymbol{K}_I^F$. These matrices have non-zero values on the diagonal of the axis being controlled, which in the case of the sawing task is the z-axis in Cartesian space.

The impedance controller imposes a mass-damping behaviour, which maintains a relationship between the desired position $\boldsymbol{x}_d$ and velocity $\dot{\boldsymbol{x}}_d$

$$\boldsymbol{F}_{imp} = \boldsymbol{K}\left(\boldsymbol{x}_d - \boldsymbol{x}_a\right) + \boldsymbol{D}\left(\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}_a\right), \tag{4}$$

where $\boldsymbol{x}_a$ and $\dot{\boldsymbol{x}}_a$ represent the actual position and velocity of the end-effector of the robot respectively, $\boldsymbol{K}$ is the stiffness matrix, and $\boldsymbol{D}$ is the damping matrix. The stiffness and damping matrices are diagonal matrices, which have non-zero values on the diagonal for the axes desired to be controlled. The goal of the learning process, described in Section II-C, is to learn the desired position and the corresponding stiffness matrix. The damping matrix is defined as a function of the stiffness matrix to achieve a critically damped system [33].

The control force $\boldsymbol{F}_{control}$ is defined in the Cartesian space, however, the used robots are controlled at the robot's joint torque level and therefore need to be transformed using

$$\boldsymbol{\tau} = \boldsymbol{J}_r^T \boldsymbol{F}_{control}, \tag{5}$$

where $\boldsymbol{\tau}$ are torques send to the robot and $\boldsymbol{J}_r$ is the robot arm Jacobian matrix. To account for the mass of the robot, a gravity compensation function has been implemented. However, as this is not the scope of the research, it is left out of the remaining of this paper.

B. Skill Propagation

The process of skill propagation describes how one agent passes a known skill to another agent, who then will pass it to the next, and so on. One of these steps is implemented using the three-staged learning scheme, as described in Section II-C. While this research implements the steps of skill propagation linearly (robot 1 teaches robot 2, robot 2 reaches robot 3, and so on) for the sake of a controlled examination, the reality would look like an exponentially growing tree, as shown in Fig. 1a. This means that if robot 1 has taught robot 2 the task, it will continue teaching robot 3, followed by robot 4, and so on. We decided to use the linear process of skill propagation, as we are interested in how the skill evolves over the different runs. Using the exponential concept would limit this research, as we would continuously investigate a similar step of propagation, namely robot 1 teaching another robot. In addition, this also makes the analysis of the skill mutation much more difficult, as simply making this a function of the different propagation steps would not be possible.

We decided to manually define the skill of the initial agent, consisting of the desired trajectory and corresponding stiffness. The reason for this is that this research focuses on robots transferring their skill to another, not on the ability of a human to transfer its skill to a robot. Therefore, in the rest of the paper, skill propagation will only refer to the transfer of skills between robots.

To define the desired trajectory x_d of the initial robot, we observed the movement exerted by humans in collaborative sawing from previous studies [28, 34–36], and propose a model that can replicate this behaviour. Since human sawing movements exhibit a form of a periodic sinusoidal signal, we define the mathematical model as

$$x_d(\phi) = \sqrt{\frac{1+n^2}{1+n^2 x_{sin}^2(\phi)}} x_{sin}(\phi) \frac{\Delta x_0}{2}, \tag{6}$$

$$x_{sin}(\phi) = \sin\left(\frac{2\pi}{\tau} t\right), \tag{7}$$

where $n = 0.5$ is a constant responsible for "flattening out" the sine function, $\tau = 2\,\mathrm{s}$ is the period of the signal, and $\Delta x_0 = 0.15\,\mathrm{m}$ the initial stroke displacement. The last two values were defined based on the results of [28].

In literature, it was found that the human was exerting a leader/follower behaviour allocated with the pushing/pulling motion respectively in a reciprocal manner [28, 31]. Therefore, we propose to define the stiffness as a function of the time derivative of the trajectory, the velocity $\dot{x}_d$. When the velocity is negative, a high stiffness (K_{max}) is applied, as the robot should be pulling. When the velocity is positive, the robot should push so the stiffness is set to zero, which means the robot is compliant

$$K = \begin{cases} K_{max} & \text{if } \dot{x}_d \leq 0, \\ 0 & \text{if } \dot{x}_d > 0. \end{cases} \tag{8}$$

C. Learning Scheme

For the robot to obtain the skill needed for the sawing task, a three-staged learning scheme based on [28] was implemented. This method assumes that there are two agents: a novice who has no knowledge of the skill and an expert who is already skilled. Both are, for this research, always assumed to be robots. The skill of the robot consists of the impedance part of the hybrid controller (4). Therefore, the goal is to learn the desired inputs of the impedance controller: the desired trajectory and the corresponding stiffness scheme. The desired trajectory is learned in stage 1 of the learning scheme, and the corresponding stiffness scheme in stage 2. As the aim is to learn a periodic sawing movement, the desired trajectory and stiffness can be defined as a function of the phase $\phi \in [0, 2\pi]$. This phase dictates the current progress of the robot during a sawing movement. For example, at the starting position of the motion (x = $0.0\,\mathrm{m}$) $\phi = 0\,\mathrm{rad}$. For the initial sawing motion, where the expert is manually programmed using (6) and (8), when $\phi = 0.5\pi\,\mathrm{rad}$ the saw is closest to the expert and when $\phi = 1.5\pi\,\mathrm{rad}$ the saw is closest to the novice. Once the phase reaches $2\pi\,\mathrm{rad}$, the saw returns to its original position, and the cycle begins again. A full sawing movement from $\phi = 0\,\mathrm{rad}$ to $\phi = 2\pi\,\mathrm{rad}$ is referred to as one period. An example of an implementation of this learning scheme has been provided in Fig. 3. In this example, Fig. 3a indicates the current stage of the learning scheme.

The goal of the *first stage* is to learn the desired trajectory. During this stage, the novice robot will be compliant, meaning its stiffness equals zero. In contrast, the expert robot will have a high, constant stiffness, which allows it to be the leader and thus control the movement. As both robots are holding the saw, an interaction force will result in the novice robot following a similar trajectory as the expert. This has been visualised in the Fig. 3b, where it could be observed that the measured trajectory of the novice is almost identical to the negative projection of the expert's measured trajectory. To obtain the desired trajectory, the position of the end-effector of the novice is measured. This measured trajectory in the first stage will be encoded as the desired trajectory for the second stage using the encoding method described in Section II-D.

The goal of the *second stage* is to learn a stiffness scheme by encoding the desired stiffness. The expert robot is controlled using its known skill consisting of a desired trajectory and stiffness scheme, in other words, its known impedance controller. The novice robot will use its desired trajectory, learned in the previous stage, and the measured trajectory to compute a stiffness scheme. In this stage, the stiffness used to control the motion of the novice robot is the desired stiffness instead of the encoded stiffness. This is because results have proven that the learning algorithm can provide quite large values for the stiffness initially, which could mean that the behaviour becomes unsafe.

In this research, the desired stiffness K_d is computed using a continuous function. This function is based on the difference between the desired x_d (learned in stage 1) and actual x_a position (measured) of the end-effector of the robot,

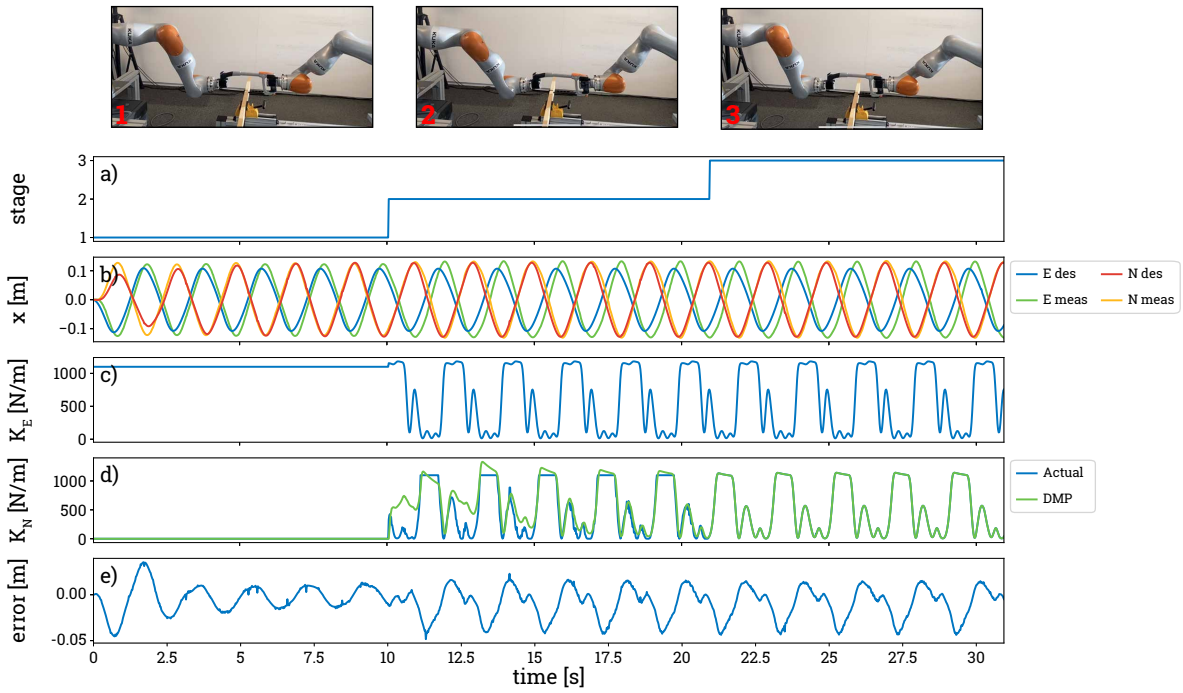


Fig. 3. Example of a three-staged learning scheme, used for the propagation of a skill from one robot to another. The sequence of images above the graph shows the movement of the robots during the three different stages. a) Shows the stage of the learning process as described in Section II-C. b) Shows the desired x_d and the measured x_a trajectory of both the expert (E) and novice (N) robot, which is learned using DMP and LWR. c) Shows the stiffness of the expert robot. d) Shows the actual stiffness of the novice, used to control the motion of the novice robot, and the encoded stiffness by means of DMP. In stage 2, the novice robot is controlled using the desired stiffness computed using (9), and in stage 3, the novice robot is controlled using the encoded stiffness, making the actual and the encoded stiffness the same. e) Shows the error between the measured and desired position of the novice robot, based on which the desired stiffness is computed (9).

$$K_d(\phi) = \begin{cases} \left(\frac{|e(\phi)|}{e_{th}}\right)^2 K_{max} & \text{if } |e(\phi)| > e_{th}, \\ K_{max} & \text{if } |e(\phi)| \geq e_{th}, \end{cases} \tag{9}$$

$$e(\phi) = x_d(\phi) - x_a(\phi) \tag{10}$$

where e_{th} is the threshold of the error, and K_{max} the maximum stiffness value. The maximum stiffness has the same value as the stiffness used by the expert in stage 1. Previous work [28] implemented a discrete function K_d that was set to either the maximum value or zero depending on whether the error exceeded or was below the threshold, respectively. In this research, the filtering done by the encoding method resulted in the discrete reference stiffness becoming continuous. It is possible that the filtering gives bad results. As we cannot control this resulting stiffness, we decided that using a continuous scheme would be more suitable. A comparison of the effects of both schemes can be found in Appendix A-A.

The *third stage* is used to demonstrate the learned behaviour, using the learned trajectory and stiffness obtained in the previous two stages. During this stage, the novice and the expert robot will use their known trajectory and stiffness scheme as inputs for their impedance controller. Before this stage, the novice robot has completed the learning, meaning that two experts are now working together to execute the task.

We discovered that a shift in the stiffness scheme occurred in some cases when transitioning from the learning stage (stage 2) to the execution stage (stage 3). This shift increased or decreased the average value of the entire stiffness scheme. It occurred when the time derivative of the stiffness was relatively high. An example of this shift can be seen in Appendix A-B. To prevent this, we propose implementing a safety regulation that requires the absolute slope of the stiffness not to exceed 15 for each of the past five measurements.

D. Skill Encoding

Periodic Dynamic Movement Primitive (DMP) [37–39] was selected to encode the skill of the robot. DMP allows for online encoding of trajectories following a rhythmic motion, making it a suitable choice for the periodic sawing task. The difference compared with the discrete variant of the DMP is that the discrete variant only allows for point-to-point learning and is therefore not useful for the sawing task.

The DMP for a single DoF trajectory y, is defined by a set of nonlinear differential equations

$$\dot{z} = \Omega(\alpha(\beta(-y) - z) + f(\phi)), \tag{11}$$

$$\dot{y} = \Omega z, \tag{12}$$

$$\tau\dot{\phi} = 1, \tag{13}$$

where Ω is the frequency, z the auxiliary variable, α and β are positive parameters defining the behaviour of the second order system, which are set to 8 and 2 correspondingly, ϕ the phase, and τ the time of one period. To ensure a smooth

behaviour, the initial phase ($\phi = 0$) must be equal to the final one ($\phi = 2\pi$).

The forcing term $f(\phi)$ is defined with $N = 50$ Gaussian kernel functions ($\Psi_i(\phi)$), which allow for a smooth trajectory

$$f(\phi) = \frac{\sum_{i=1}^{N} \Psi_i(\phi) w_i r}{\sum_{i=1}^{N} \Psi_i(\phi)}, \tag{14}$$

$$\Psi_i(\phi) = \exp\left(h\left(\cos\left(\phi - c_i\right) - 1\right)\right), \tag{15}$$

where c_i are the centres of the Gaussian basis functions distributed along the phase of the movement, h_i their widths which have been set to 0.05, and w_i the weights to be learned. The parameter r can be used to modulate the amplitude of the periodic signal. If this is not used, which is the case for this research, it can be set to $r = 1$ [40].

To learn the forcing term $f(\phi)$, one could invert (11) and use a measured demonstrated trajectory $y_d(t_j)$ and its time derivatives $\dot{y}_d(t_j)$ and $\ddot{y}_d(t_j)$. In the case of the sawing task, this demonstrated trajectory is the actual pose of the end-effector when learning the desired trajectory in stage 1 or the computed stiffness when learning the desired stiffness in stage 2 (Section II-C). This result in the desired shape f_d to be approximated by

$$f_d\left(t_j\right) = \frac{\ddot{y}_d\left(t_j\right)}{\Omega^2} - \alpha\left(\beta\left(-y_d\left(t_j\right)\right) - \frac{\dot{y}_d\left(t_j\right)}{\Omega}\right). \tag{16}$$

In order to learn the weights w_i of the DMP, a Locally Weighted Regression (LWR) is used, as it is suitable for online learning due to its ability to quickly update the model [28, 39]. This method update the weights using recursive least-squares method, which is based on the error e_r ((18)) between the desired trajectory shape f_d and the currently learned trajectory shape, as well as a forgetting factor $\lambda = 0.995$ which determines the rate of weight change

$$w_i\left(t_{j+1}\right) = w_i\left(t_j\right) + \Psi_i P_i\left(t_{j+1}\right) r e_r\left(t_j\right), \tag{17}$$

$$e_r\left(t_j\right) = f_d\left(t_j\right) - w_i\left(t_j\right) r, \tag{18}$$

$$P_i\left(t_{j+1}\right) = \frac{1}{\lambda}\left(P_i\left(t_j\right) - \frac{P_i\left(t_j\right)^2 r^2}{\frac{\lambda}{\Psi_i} + P_i\left(t_j\right) r^2}\right). \tag{19}$$

The initial values of the parameters are set to $P_i(t_0) = 1$ and $w_i(t_0) = 0$. Once learning is complete, the learned weights can be saved in order for the DMP to be used at a different time, for example, during the next skill propagation step.

We conducted an experiment to validate the values used for the parameters of the DMP and LWR. This has been described in Appendix B.

III. Experiment & Results

In this chapter, we present the experiments conducted and their results. The first section will entail a description of the experimental setup and the protocol used. This is followed by a description of the metrics employed to analyse the results: the peak-to-peak amplitude/stroke displacement, the travelled distance, the midpoint of movement, the average value/stroke offset, and the number of peaks. In Section III-C, the results of the experiments using the default settings will be presented, followed by the results of changing the external factors (maximum stiffness, configuration, period, and friction coefficient) in Section III-D.

Initially, the task was implemented in the real world. We found that one skill propagation step (also referred to as one run) would take roughly 5 minutes. This time included mounting the object, setting the robots into the correct initial position, launching the control, and carrying out the collaborative sawing task. We decided to carry out 30 steps for each tested setting, as the results$f(\phi)$, using the default settings, showed that the most interesting mutation features occurred during the first 20 runs. Therefore, 30 runs would allow enough time to evaluate the most interesting mutations. We refer to these runs as one trial, which would take approximately 2.5 hours to execute. As we are also interested in the reproducibility of the mutation, we decided that each trial should be executed 5 times. Therefore, this experiment would take approximately 12.5 hours. However, these 12.5 hours only give results for one parameter setting. As we are also interested in how changing these settings would affect the mutation, multiple additional trials should be conducted. In total, we tested 20 different settings, which in the real-world would take approximately 250 hours. In addition, the real-world setup also requires an object to be sawn for the 30 runs $\times$ 5 trials $\times$ 20 settings $=$ 3000 different propagation steps. Besides these time and resource limitations, another disadvantage is that it could not be promised that skill propagation would not lead to unsafe behaviour. This unsafe behaviour has a high potential of arising after multiple propagation steps, as each step allows for the skill to deviate further, resulting in a skill that can be quite different from the initial one. Therefore, we decided to execute the runs in simulation. An additional advantage of using a simulation is that the external factors can be fixed, which allows for a more controlled investigation of their influence on the mutation. *Gazebo* is used to simulate the robots interacting with their environment, and *Robot Operating System (ROS)* as a tool to control the robots. This setup is described in Appendix C.

A. Experimental Setup & Protocol

Multiple experiments have been conducted to investigate the mutation during skill propagation. During each of these experiments, the goal was the same; The propagation of the sawing skill from the expert to the novice robot, using the three-staged learning scheme which will be presented in Section II-C. One execution of the skill propagation using this learning scheme is referred to as a run. Each stage of the learning takes $10\,\mathrm{s}$, resulting in a total time of $30\,\mathrm{s}$ per run. The entire skill propagation process (robot 1 teaching robot 2, teaching robot 3, etc.), is called a trial. In this research, we conducted $5\,\mathrm{trials}$ per experimental setting, each consisting of $30\,\mathrm{runs}$.

The experiments are split into two groups: the group using the default setting of the external factors, and the group with changed values for the external factors. The external factors tested are presented in Table I, which also shows their default settings. Each factor was changed individually to investigate

TABLE I
THE EXTERNAL FACTORS TESTED, AND THEIR DEFAULT SETTINGS.

Factor	Variable	Value
Maximum stiffness	K_{max}	$1100\,\mathrm{N\,m^{-1}}$
Base position	-	Both at y= $0.4\,\mathrm{m}$
Period	τ	$2.0\,\mathrm{s}$
Friction coefficient	μ	0.0

whether changing these values would influence the mutation, and if so, how. This meant that each factor would be set to their default value during this group of experiments, except the one being investigated.

The default setting is used for three purposes: 1) To investigate whether mutation occurs and, if it occurs, why. 2) As a baseline, which is used to compare the influence of the changed external factors. 3) To show whether the mutation is repeatable.

For the external factor group, we only set one factor to a non-default value during each trial. This allows the isolation of the factors, resulting in a controlled examination of their effect on skill mutation. By comparing the results with the default setting, we can investigate which factors do or do not influence the skill mutation. If no difference is observed, we can assume that the effect on the overall mutation is minimal. As for the default setting, each changed factor is tested using 5 trials. An example of a changed external factor, is the time of one period (τ). In addition to the default settings, we investigated two different settings (τ=$1\,\mathrm{s}$ and τ=$3\,\mathrm{s}$), resulting in a total of 10 trials. Besides the four factors shown in Table I, six additional external factors were tested; however, they have proven to be less significant and are therefore discussed in Appendix D.

We implemented sawing experiments with the KUKA LBR iiwa7 and KUKA LBR iiwa14 robots in the real world and a simulation environment. From now one, these robots will be referred to as the iiwa7 and iiwa14, respectively. To control the robots, we used ROS, an open-source robotics framework that enables robots to interact with their environment [41]. An overview of the control structure is provided in Appendix C-A. As the real-world and simulated setups have different requirements, they are discussed separately.

For the real-world implementation, we used a BAHCO metal saw featuring handles on two sides so that both robots could hold it. The saw is attached to the robot with a 3D-printed clamp, which is mounted to the robot. The object to be sawn is a wooden plank. A visualisation of this setup has been provided in Appendix C-B.

We simulated the motion and interaction of the robots using *Gazebo*. The saw was modelled as a rectangular object that weight was distributed equally along its length. It was attached to the robot's end-effector through a virtual joint, preventing it from dropping. The object that was sawn had a size of $0.565\,\mathrm{x}\ 1.0\ \mathrm{x}\ 0.1\,\mathrm{m}$, with a grove in the middle along the x-axis with a depth of $0.3\,\mathrm{m}$ and a width similar to the saw. This constraint the movement of the saw along the y-axis on both sides and towards the bottom along the z-axis. We did not consider that parts of the object would be sawn away, resulting in an increased depth of the grove. Instead, this depth remained the same throughout the sawing motion. This was done as this research did not focus on the robot's ability to saw an object. A visualisation of this setup has been provided Appendix C-C.

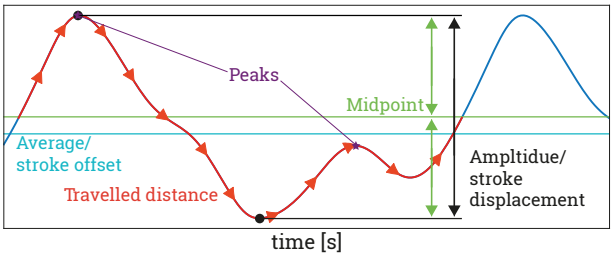


Fig. 4. Graphical representation of the used metrics. The metrics are defined as follows: peak-to-peak amplitude/stroke displacement (difference between maximum and minimum value), travelled distance (total distance of trajectory), the midpoint of movement (between maximum and minimum value), average value/stroke offset, and the number of peaks.

B. Metrics

To analyse both the skill of the robot (desired trajectory and stiffness) and outputs (e.g., forces on end-effector, joint torques, and joint angles) we defined multiple metrics, of which a graphical representation has been provided in Fig. 4. As we are interested in the evolution of the skill throughout the different runs, we defined these metrics as a function of the different runs. To do so, we computed the value per metric over the span of one period. If the signal to be investigated consisted of a time longer than one period, we took the average per period as a result. For example, if one is to investigate the desired trajectory during stage 1, then the result would be the average of $10\,\mathrm{s}/(\tau{=}2.0\,\mathrm{s})$ = 5 periods. As discussed, each experiment consists of 5 trials. Therefore, the results are shown as the average of the five trials (per run) and their standard deviation. This additionally allows for the investigation of the repeatability of the skill, as a low standard deviation means that the main trends occurring in the different mutations are similar.

Peak-to-peak amplitude/stroke displacement: The distance between the maximum and minimum value of a signal over one period is called the peak-to-peak amplitude $|\max(x(\phi)) - \min(x(\phi))|$. When discussing the measured or desired trajectory, this metric is referred to as the stroke displacement.

Travelled distance: The distance travelled by the end-effector of the robot in during one period $\int_0^{2\pi} |\delta x(\phi) d\phi|$. This metric is only used to investigate the trajectory and does not have a general name.

Midpoint of the movement: The point halfway between the maximum and minimum value $(\max(x(\phi)) + \min(x(\phi)))/2$ As this metric is only used to investigate the trajectory, no general name exists. In the case of the default settings, the base of the iiwa7 and iiwa14 are located at $x = -0.7\,\mathrm{m}$ and $x = 0.7\,\mathrm{m}$, respectively. A value for the movement's midpoint equal to $0.0\,\mathrm{m}$ would mean that the midpoint was right between these robots. An increase in this value means that the midpoint is shifted towards the iiwa7, whereas a decrease means it is shifted towards the iiwa14.

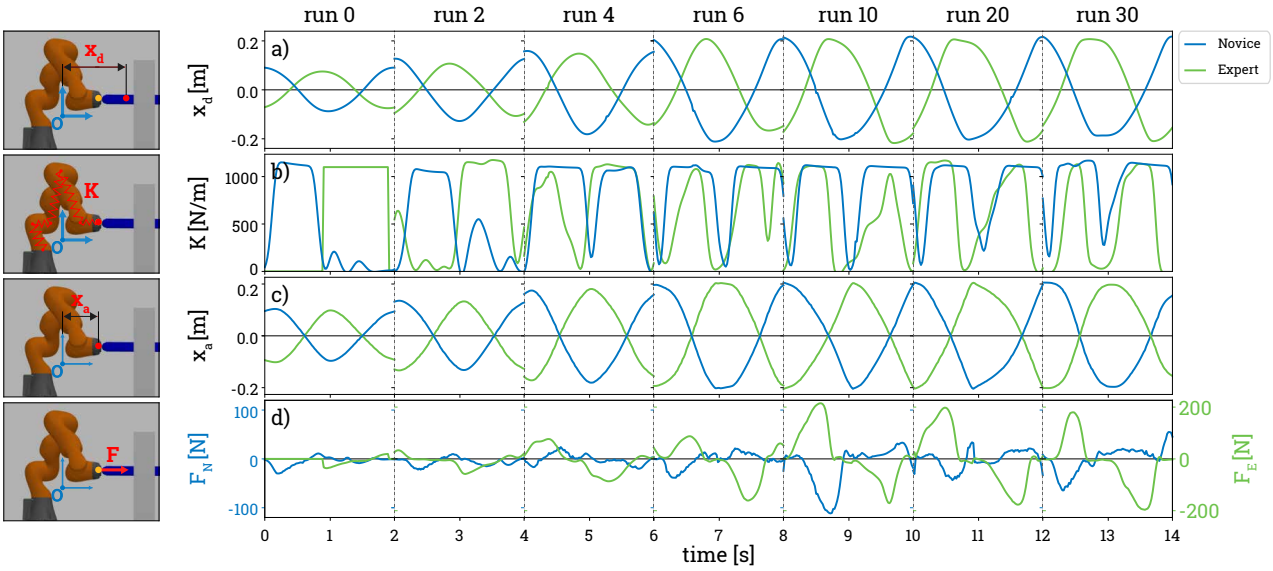


Fig. 5. The skill evolution of the expert and novice robots showed using one period of stage 3 of the learning scheme for different runs. On the left, four images describing the different component has been visualised. In the graph, from left to right, different runs are shown from one trial. Note here that the runs are not linearly spaced from 0 to 30. Each of the graphs a)-d) show both the results of the novice (blue) and expert (green). From top to bottom; a) Visualises the desired trajectory x_d, here means a high value that the end-effector of the robot is further away from the robot's origin. b) Shows the stiffness scheme. c) Illustrates the actual trajectory x_a of the robots. d) Shows the forces sent to the robot, where the left axis is used to show the forces of the novice robot and the right axis the forces of the expert robot.

Average value/stroke offset: The average value of the data $(t_{2\pi} - t_0)^{-1} \int_0^{2\pi} x(\phi) d\phi$. It should not be confused with the average named before, which describes the average per run. When discussing the measured or desired trajectory, this metric is referred to as the stroke offset. Just as for the midpoint of the movement, a value higher and lower than $0.0\,\mathrm{m}$ indicate that the movement is closer to the iiwa14 and iiwa7, respectively.

Number of peaks: The number of peaks illustrates the complexity of the data, meaning a large number of peaks states that the signal is more complex than a low number. Note that for stiffness, a flattened peak is often found where the maximum stiffness was reached. These "flattened" peaks are counted as only one.

Besides these objective metrics, we also analysed the data qualitatively by using the raw data to investigate the evolution of the skill. In contrast to the previously discussed metrics, the raw data is a function of time rather than a function of the different runs.

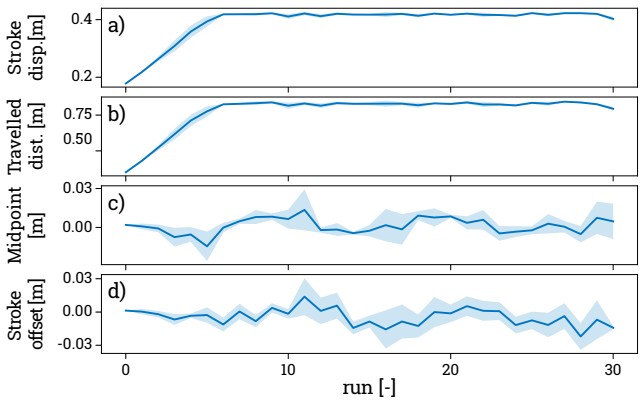


Fig. 6. Analysis of the desired trajectory learned by the novice robot in stage 1 of the learning scheme. Different metrics are used for this analysis. a)-d) Illustrate the stroke displacement, the travelled distance, the midpoint of the movement, and the stroke offset, respectively.

C. Default Settings

In this section, the mutation of the skill of a robot will be discussed using the default settings as defined in Table I. A quantitative analysis of the mutation in skill of both the expert and novice robot, has been visualized in Fig. 5. This graph shows one period of the learned skill (stage 3) for multiple runs of one trial.

The desired trajectory x_d (Fig. 5a), learned in stage 1, shows an increase in stroke displacement throughout from run 0 to run 6, after which it stagnates. This stagnation is confirmed when looking at the analysis of the stroke displacement in Fig. 6a. The stagnation of the stroke displacement results from reaching the environment's boundaries, i.e. the saw is touching the object it is sawing. This limit is at $0.45\,\mathrm{m}$, which is equal to the length of the saw. There is a similar pattern visible for the distance travelled (Fig. 6b). Based on these metrics, we could assume that the desired trajectory is smooth with only one peak. This is substantiated by looking at Fig. 5a.

The results of the midpoint and the stroke offset of the desired trajectory (Fig. 6C and D) show similar patterns. Based on this and the results in Fig. 5, it can be concluded that the general motion characteristics in one direction are similar to the other direction. If these were different, the stroke offset would be shifted compared to the midpoint of the movement. Upon further analysis, it is evident that the stroke offset has a zigzag pattern; In the even-numbered runs, the stroke offset

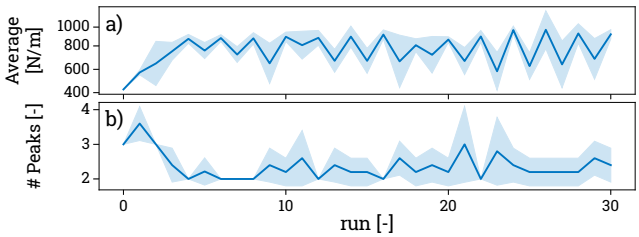


Fig. 7. Analysis of the stiffness learned by the novice robot in stage 2. Different metrics are used for this analysis. a) Shows the average value of the stiffness scheme. b) Shows the number of peaks per period.

moves down (compared to the prior run), which indicates that the movement is shifted towards the iiwa7 robot. Moreover, in the odd-numbered runs, the stroke offset moves up, which indicates that the movement is shifted towards the iiwa14 robot. As the role of the expert robot is switched between the two with each run, we found that the movement is always directed towards the expert robot.

Analysing the stiffness K of the robots (Fig. 5b) shows that initially, there was a pattern of high stiffness when pulling and low stiffness when pushing, which is logical as this is the manually defined input we gave to the expert in the first run ((6) and (8)). This pattern, however, starts to disappear in the following runs (run 2) and has eventually completely disappeared (run 4). By analysing the stiffness learned by the novice in stage 2 (Fig. 7), we see that the stiffness's average value increases throughout the different runs. The raw data showed that in the later runs, the error between the desired and measured position of the robot increased. As the computed stiffness depends on this error (9), the stiffness was often set to its maximum value. Therefore, peaks only occurred when the desired trajectory crossed the measured trajectory, which for most cases was only twice. The complexity of the stiffness will be further discussed in Section III-D.

The task execution, a result of the learned skill (desired trajectory and stiffness) and the collaboration between the robots, has been presented in Fig. 5c by means of the measured position. Comparing the desired and measured positions shows that they are different. This is partial because in stage 3, both robots try to control the movement, whereas in stage 1, when the desired trajectory was learned, only the expert controlled the robot's movement. Another reason has been indicated in Fig. 3b, which shows the difficulty for the expert robot to match the desired trajectory in stage 1. This results from the impedance control law (4), which requires a difference between the desired and measured trajectory to compute a force. In runs 10 and 20 (Fig. 5), we see that this results in the peaks of the measured trajectory being sharper than those of the desired trajectory. These sharp peaks have been smoothed out in run 30. Based on this, we can conclude that if the robot cannot overshoot on the desired trajectory, which is especially done in the earliest runs, as a result of the boundary limits, it can smooth out its trajectory.

The forces used to control the robot along the x-axis, the direction of movement for the sawing task, are visualised in Fig. 5d. These show that the absolute maximum force used by the expert is almost double the one used by the novice (note that different scales are used for the forces of the robots). The difference between these forces indicates that the influence of the expert on the movement is twice as large as the one of the novice. As both forces are computed using (4), this difference can either result from a larger value for the stiffness, which we can see in Figs. 5 and 7 is not the case or a larger error between the desired and measured trajectory. We already identified that a phase lag between the desired and measured trajectory always occurred, even in stage 1. As the novice robot learns from a similar trajectory as the expert is exerting in stage 1, it will learn from this phase-lagged trajectory and will, therefore, always have an equal or smaller lag than the expert in stage 3. As this lag largely influences the error, the expert's resulting forces will be larger than those of the novice. In Section III-D, we will elaborate on how the size of this lag influences mutations of the skill. In addition to the phase lag, the increased force is also a direct result of the increased travelled distance (Fig. 6b). As the travelled distance increase by a factor of 2, it is reasonable that so do the forces. However, the increasing forces are approximately increased by a factor of 4. Therefore, this increase is not solely dependent on this increased travelled distance, but rather a combination of this increase and the phase lag. Looking at the size of these resulting forces in the final run, we see that the forces are quite large, which can result in unsafe behaviour. We discuss this possibility of unsafe skills due to large forces in Section IV.

One could also discuss the skill (desired trajectory and stiffness) as one aspect instead of two. The initial run shows a clear distinction between when the novice is putting in the effort and when the expert is. They are both, at times, exerting high stiffness when pulling and lows stiffness while pushing. This clear distinction, however, fades throughout the different runs. At the same time, the size of the force increases. This increase in force would let one think that both robots are trying to pull or push, which would result in both forces increasing. However, the results show that the forces are both in the same direction and amplify each other. While this amplification is visible, it is evident that the more significant influence on the movement is due to the expert's contribution, which was already identified as a result of the phase lag.

D. Influence of the External Factors

The second set of tests was conducted to show how different external factors influence mutations. We tested ten external factors, of which four showed significant differences compared to the default settings described in Section III-C. The factors which showed this significant influence on the mutation will be discussed in this section. These factors are the maximum stiffness, the length of the period, the base position of the robots, and the friction coefficients. Their default settings were presented in Table I. Appendix D contains a analysis including the six additional tested factors.

For each of the tested factors in this section, it was visible that mutation occurred. We conducted a general analysis, of which the results have been presented in Table II. This analysis consists of the following aspects; Whether the resulting

TABLE II
COMPARISON OF THE EXTERNAL FACTORS.

Factor	Symbol	Default value	Value	Same as default	Repeatable[1]	Converges
Maximum stiffness	K_{max}	1100 N	500 N	No	Yes	Yes[2]
			5000 N	No	Yes	Yes[1]
Period	τ	2 s	1 s	No	No	Yes[2]
			3 s	No	No	No
Base position	-	Both at y = 0.4m	Both at y=0.0 m	No	Yes	No
			y=0.4 m/y=-−0.4 m	Yes	Yes	Yes[2]
			y=0.4 m/y=-0.5 m	Yes	Yes	Yes[2]
Friction coef.	μ	0.0	0.05	No	Yes	Yes
			0.01	No	Yes	Yes[2]
			0.005	No	Yes	Yes[2]

[1] Repeatable is set to "yes" if the main trends of the different trials are similar. As will be discussed in Section III-E, the mutations are never repeatable throughout the different trials, but often their main trends are.
[2] This mutation converges due to approaching some limit, i.e. an environmental boundary, a joint limit or a torque limit.

mutation was the same or similar as the mutation occurring for the default settings (Section III-C). If the mutation was repeatable, which is determined by comparing the five different trials tested. Here we state that a mutation is repeatable if the main trends are. And whether the mutation converges, either due to approaching a limit or due to some stable behaviour.

The remaining section will discuss the mutation patterns found when analysing each of the individual factors. It was chosen not to discuss the tested factors individually, as often different tested factors resulted in similar patterns. The patterns discussed are the influence of the phase lag, overshoot on the desired trajectory, effect of the joints states, and approaching torque limits.

1) Phase Lag

In the example of a learning scheme, provided in Fig. 3, it is visible that the desired trajectory of the novice and the expert are not identical. This difference is a result of the inability of the expert robot to match the desired trajectory perfectly. We define this difference between the desired and measured trajectories as the *phase lag*. This phase lag does not necessarily have to be an issue; if the measured trajectory matches the desired trajectory, according to the impedance law there would be no force to move the saw (4). However, we see in the results that, besides a shift of the phase, the phase lag also changes the characteristics of the skill. As the novice robot is connected to the expert robot by means of the saw, these changes in characteristics will also influence the learned skill of the novice robot.

In Fig. 8a, three cases are presented where the phase lag causes alterations in the characteristics of the learned trajectory. The example using the default settings (Fig. 8a.1) and the one using a low value for the maximum stiffness K_{max} =500 N (Fig. 8a.2) both show an overall increase in the stroke displacement throughout the different runs. This is confirmed when looking at the visualisation of the analysed results in Fig. 8b.1. As a result of the phase lag, forces are computed to control the movement, which allows the robot to overshoot on the desired trajectory. When the desired trajectory and the measured trajectory intersect, the force will switch direction. However, we observe that these resulting forces are not strong enough to counteract the initial velocity

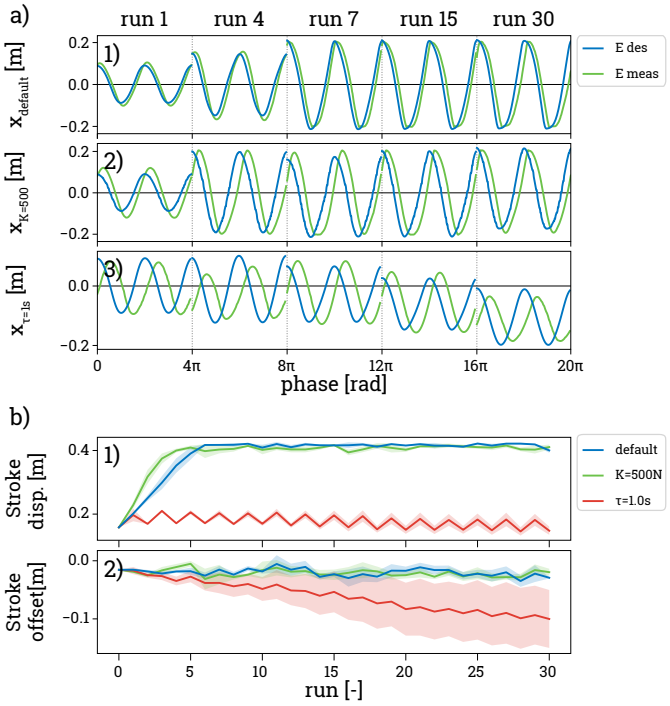


Fig. 8. Influence of the phase lag on the mutation of the learned trajectory. a) Shows from left to right the desired and measured trajectory of the expert during the second and third period of stage 1 of different runs. From top to bottom, the default setup (1), a setup where the maximum stiffness was changed to K_{max} =500 N (2), and a setup where the period was set to 1 s is visualised (3). b) Shows the evolution of the stroke displacement (1) and the stroke offset of the learned trajectory of the novice robot during the different runs (2).

in the opposite direction. This, again, does not necessarily have to be an issue, as we see, for example, in run 4 of the 500 N case (Fig. 8a.2), that this intersection often occurs when the peak of the desired trajectory is already passed. So, to reach the same stroke displacement as the desired trajectory, a small amount of overshoot on the desired trajectory is required to maintain the same trajectory characteristics. However, we see that the resulting peaks of the measured trajectory exceeded the ones of the desired trajectory, resulting in an overshoot of the stroke displacement. By comparing the default and low

maximum stiffness cases, we see that the overshoot is less in the case of the smaller lag. For a smaller lag, it will thus take more runs until the boundaries of the environment (at $0.45\,\mathrm{m}$) are reached.

In Fig. 8a.3, we showed a case ($\tau = 1\,\mathrm{s}$) with a larger phase lag than previously discussed. This phase lag can result in the inability of the robot to match or exceed the stroke displacement (e.g. run 4 and 30). However, this was not always the case, as we also identified cases where the robot exceeded this displacement (e.g. run 7 and 15). We found that the stroke displacement (Fig. 8a) follows a zigzag pattern, which matches the zigzag pattern of the stroke offset (Fig. 8b). This indicates that the expert robot always moves the entire movement slightly closer to its base position. When the iiwa7 robot is the expert, the entire movement shifts more towards its base than when the iiwa14 is. As will be discussed in Section III-D3, this can be attributed to an average decrease in the stroke offset.

Besides the trajectory, the phase lag also influences the learned stiffness in stage 2. As the stiffness is calculated using the error between the desired and measured trajectory (9), a larger gap between these trajectories should result in higher stiffness on average. To confirm this, three different values for the maximum stiffness have been compared, the results of which are provided in Fig. 9a. Note that we used a normalized average stiffness instead of the actual ones, as this allows for a more useful comparison between the stiffness characteristics. The results show, that when the used stiffness is low (K_{max} =500 N), the phase lag increases, resulting in a larger error between the desired and measured trajectory than for the cases with high stiffness. This larger error is reflected in the computed size of the stiffness (9), which results in a higher normalised average stiffness than the case with high stiffness (K_{max} =5000 N), when the phase lag is low (as is visible in Fig. 9b.1). Additionally, (9) states that if the error between the desired and measured trajectory exceeds a set threshold, the stiffness is set to the maximum value. If this threshold is exceeded for an extended time, a flattened part in the stiffness will be visible, which is the case for a low stiffness (500 N) in Fig. 9a.3. As a flattened peak will only be counted as one in the complexity analysis (Fig. 9b.2), the results of a low stiffness will show a lower amount of peaks than the cases with a higher stiffness. Based on the results of Fig. 9a and b, we conclude that a smaller phase lag will result in a more complex stiffness scheme, while the average value of this scheme will be lower.

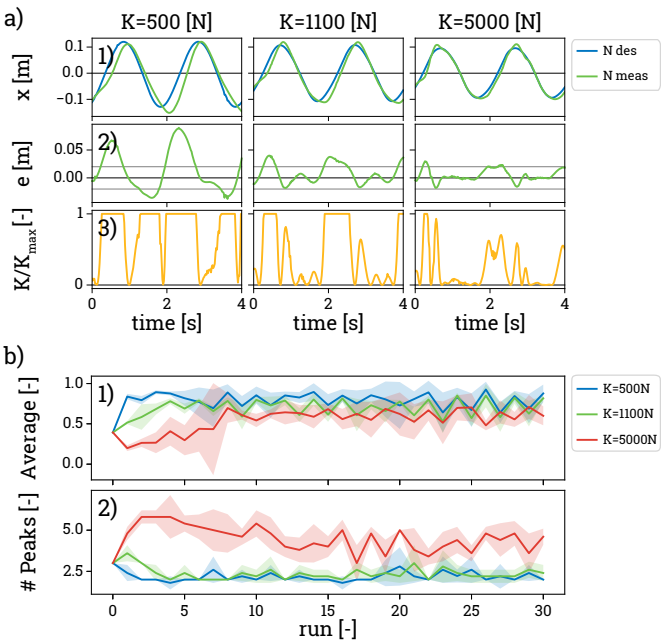


Fig. 9. Influence of the phase lag on the mutation of the learned stiffness. a) Shows two periods of the measured and desired trajectory of the novice robot in stage 2 and its normalized desired stiffness (stiffness divided by maximum stiffness). This is shown for different maximum stiffness values during run 8. b) Shows the evolution, throughout the different runs, of the normalized average value and the number of peaks of the learned stiffness scheme of the novice robot during one period.

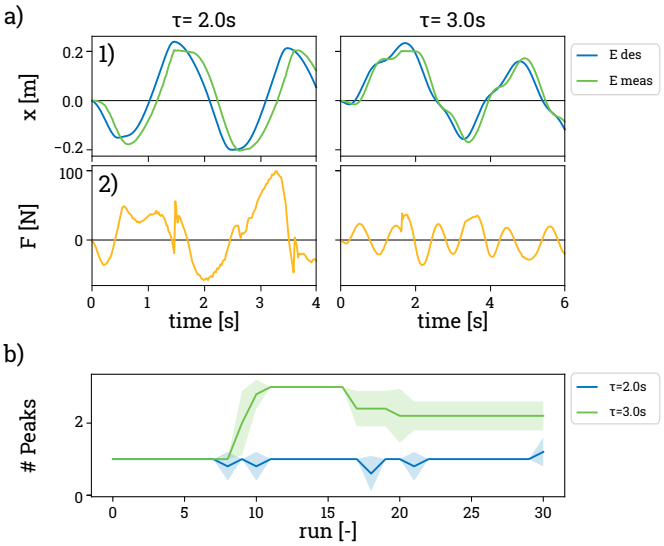


Fig. 10. Influence of overshoot on the desired trajectory on the measured trajectory. a) Shows an example of the results of two phases of stage 1 of the desired and measured trajectory (1) and its corresponding force along the x-axis (2) for the expert robot using two different periods (τ=2 s and τ=3 s). b) shows the complexity of the trajectories learned by the novice throughout the different runs.

2) *Overshoot on the Desired Trajectory*

When the error between the desired and measured trajectory is small, the robots overshoot on the desired trajectory. We provided an example of this in Fig. 10a. This shows two periods of the desired and measured trajectory, and the corresponding forces in stage 1 for both the default settings (τ=2 s) and a longer period (τ=3 s). Relating the force to the movement, we identify the results of these default settings to be primarily intuitive; Negative while pulling and positive while pushing. This is not applicable for the case using a longer period. As visible in the example, the small error between the desired and measured trajectory often results in the expert robot overshooting the desired trajectory. The robot compensates for this overshoot by applying a counterforce, resulting in a rough trajectory, i.e. more peaks. As the novice learns from this measured trajectory, a trajectory including these irregularities

will be learned. The extent of these irregularities increases throughout the different runs, which is shown by analysing the peaks of the learned trajectory (Fig. 10b). This indicates that the trajectory becomes rougher throughout the different runs.

A benefit of the mutations, resulting in a rough trajectory, can be found when looking at energy efficiency. This efficiency is computed using the work $W = \int K(x_d - x_a)dx$. As the examples provided in Fig. 10a have different periods, we decided to compare them by computing the work per period. The results for $\tau = 2\,\mathrm{s}$ is $33.3\,\mathrm{kJ/period}$, whereas it was $11.3\,\mathrm{kJ/period}$ for $\tau = 3\,\mathrm{s}$. These results indicate a trade-off between the smoothness (low complexity) of the trajectory and its energy efficiency.

3) Joint States

Testing different setups of the robot, i.e., different base positions, has highlighted the effect of joint states on the mutation. The base positions were defined as a displacement from the x-axis, which is the axis of the sawing movement. For the default setting, the robots were displaced $0.4\,\mathrm{m}$ parallel to this axis, which means that they are located at $y = 0.4\,\mathrm{m}$. We also tested a setup where the base was aligned with the x-axis, meaning $y = 0.0\,\mathrm{m}$. For both of these setups, one robot was set at $x = 0.7\,\mathrm{m}$ (iiwa14) and one at $x = -0.7\,\mathrm{m}$ (iiwa7). The sawing motion and the resulting movement of the end-effectors were both along the x-axis. Therefore, the different base positions caused the initial joint states of the robots to vary. A visualisation of the analysis, comparing these two base positions, has been provided in Fig. 11a.

In Section III-C, we already identified the converges of the default setting ($y = 0.4\,\mathrm{m}$) as a result of the boundaries of the environment. For the case where the base position of the robot was aligned with the sawing movement ($y = 0.0\,\mathrm{m}$), we see that the stroke displacement did not increase (Fig. 11a). To make sure this was not a result of a joint limitation, we tested moving the end-effector of the robot closer to itself. This showed us that it should be able to increase the stroke displacement. This was also confirmed as the stroke offset of the movement decreased while the stroke displacement was averagely stable.

The rapid decrease in the stroke offset of the movement indicates a shift towards the base position of the iiwa7 robot. However, this is not a smooth decline, but instead, a zigzag pattern is visible. This pattern corresponds to which of the two used robots is the expert; In case the iiwa7 is the expert, the stroke offset moves down, which means that the movement is shifted towards the base position of this robot. When the iiwa14 is the expert, the stroke offset moves up, shifting the movement toward this specific robot's base position. The shifting of the movement is a result of the robots' force manipulability [42], an example of which has been provided in Appendix F. The force manipulability of a robot greatly impacts the robot's ability to change directions and is dependent on the robot's joint states. This tells us that the reason the results of the $y = 0.0\,\mathrm{m}$ and $y = 0.5\,\mathrm{m}$ setup are quite different, is due to the states of the joints, which are different for the initial configurations of both robots. If the

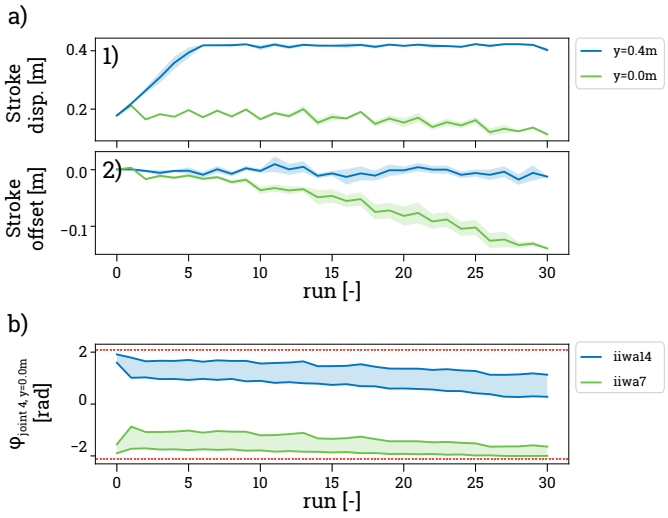


Fig. 11. Effect of joint states of mutation of robot's skill. a) Shows the stroke displacement (1) and stroke offset (2) of the robot using two different base positions of the robots. b) shows the evolution of the moving range of joint 4 by both used robot, during one trial.

end-effector is further away from the robot's base position, its force manipulability will be increased, thus enhancing its directional flexibility. When the robot is shifting from pushing to pulling, the end effector is far from the base position and the force manipulability is relatively high, enabling a successful shift in direction. However, when the robot is shifting from pulling to pushing, which occurs closer to the base position, it has a lower force manipulability, making it difficult to shift directions. If the robots are executing the task collaboratively, the other robot can compensate for the difficulty in shifting from pulling to pushing. However, during stage 1 of the learning scheme, when the expert robot controls the entire movement, the inability to shift direction can result in the robot overshooting on the stroke displacement closest to its base position. As a result, the stroke offset of the movement will move towards the expert robots during each run, resulting in a zigzag pattern visible in Fig. 11a.2.

If the influence of both the robots on the movement had been the same, we would still see a zigzag pattern in the stroke offset, but there would not be an average decline visible in Fig. 11a. This indicates that the force manipulability of one robot is higher, allowing it to shift direction more efficiently, resulting in less overshoot than the other robot. We identified the iiwa14 as the robot with higher ability to shift direction. Therefore, resulting in a decline of the stroke offset away from this robot and towards the iiwa7. A similar pattern to the one shown in Fig. 11a was also visible in Fig. 10, where $\tau = 1\,\mathrm{s}$. Throughout this research, we have treated the two robots similarly, meaning we gave them similar torques to control their position. However, the decline of the stroke offset tells us that the robots are affected differently by similar torques.

The results of the y=$0.0\,\mathrm{m}$ do not show a convergence in the learned trajectory (both the stroke displacement and stroke offset are still declining). However, we found that one joint (joint 4) was approaching its limit, therefore, affecting the

mutation. In Fig. 11b, we compare the range of movement for joint 4 of the iiwa14 and iiwa7 robots. The graph displays how the joint slowly moves towards its limit (indicated in red) throughout the different runs. The results show that joint 4 of the iiwa7 reaches one of its lower joint limits at approximately run 20. After this run, we still see that the maximum value of the joint is changing. While this only considers one joint out of the robot's seven joints, this explains why we see a more rapid decrease in stroke displacement after 20 runs. More runs are required to test whether this mutation will stabilize at some point.

4) Torque Limits

Reaching torques limits restrains the robot from following the desired trajectory. This can best be demonstrated by looking at stage 1, where the motion is only dependent on the effort exerted by the expert. An example of this has been shown in Fig. 12a, where a high static friction coefficient ($\mu = 0.5$) was applied on both the object and the saw. In the initial runs (0 to 3), the robot is able to follow the desired trajectory. In run 4, we see that this is no longer the case, as torque limitations hinder the robot. As a result, the stroke displacement significantly decreases. This is followed by the trajectory eventually smoothing out, resulting in a final behaviour with zero stroke displacement (run 30).

Run 4 is highlighted in Fig. 12b, where not only the desired and measured trajectory are shown, but also the corresponding forces required to obtain this trajectory. This example illustrates how the torque limitation results in the robot's inability to follow the desired trajectory. The force indicated in red is referred to as the send force, which was computed using the impedance control law (4). The force indicated by yellow is referred to as the measured force, which was computed by transforming the measured torques executed by the robot. It can be seen that the values of the required forces are almost 20 times as large as the measured force, indicating that the robot's torque is often saturated. By comparing the forces sent and the ones measured with the resulting trajectory, it can be seen that when the torques are flattened (e.g. at t = 1.8 s to 3.0 s), the desired trajectory cannot be followed by the expert. Therefore, the torque limitation largely influences the mutation of the trajectories.

The saturation of the torque limits also affects the learned stiffness scheme. As the variability of both the desired and measured trajectory decreases, the computed stiffness scheme is largely affected as it depends on the error between both trajectories. This results in the average and the stiffness decreasing. For the final run, the average, and the variability of the stiffness are almost being reduced to zero.

E. Reproducibility

To investigate the reproducibility of the mutation, the same run of different trials, using the default settings, have been compared, as visualized in Fig. 13. The reproducibility of the mutation is defined as the ability of the robot to reproduce the same skill (desired trajectory and stiffness), during each trial. In these examples, the offset of the phase is accounted

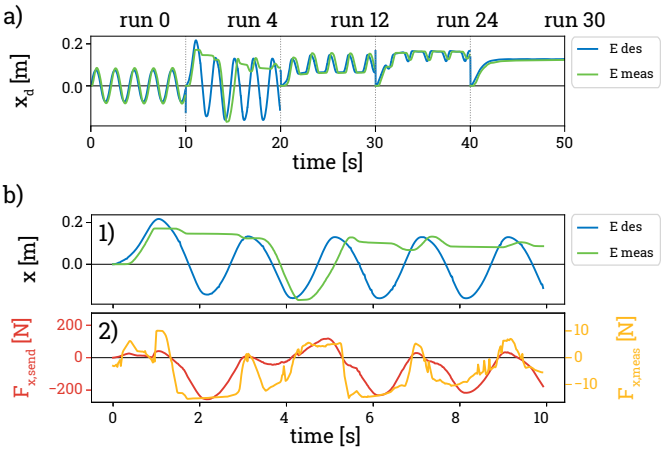


Fig. 12. Effect of torque limit on the mutation of robot's skill, illustrated with data where the friction coefficient was set to a high value. a) Shows the desired (blue) and measured (green) trajectory of the expert during the first stage of multiple runs. b) Shows stage 1 of run 2, where the upper graph (1) illustrates both the desired (blue) and measured (green) position of the expert, and the lower graph (2) the force sent to the robot (red) and the forces measured using the measured joint torques (yellow).

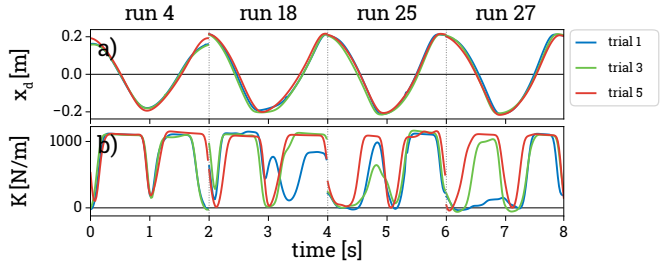


Fig. 13. One period of the skill learned by the novice robot during multiple (random) runs of different trials is shown. From left to right, four different runs are visualized. The upper graphs a) show the learned trajectory and the bottom graphs b) show the learned stiffness.

for by calculating the phase shift corresponding to a frequency component equal to $1/(\tau = 2.0\,\mathrm{s}) = 0.5\,\mathrm{s}^{-1}$. As a result, the peaks of the trajectories overlay one another.

The desired trajectory of the different trials exhibits minimal variability. However, as the trajectories are not identical, the mutation of the desired trajectory is not reproducible. Nonetheless, both Fig. 13a and the analysis of the desired trajectory in Fig. 6 reveal that the main trends of the mutation for each of the different runs are largely similar.

The differences between the stiffness scheme of the trials are much more evident. An example of this is provided in trial 1 of run 18 (Fig. 13b), where the characteristics of the learned stiffness were completely different compared to the other trials. However, for most of the other cases, we often only identified two peaks. For these cases, the valleys only appear when the desired trajectory and measured trajectory cross. The analysis of the stiffness, in Fig. 7, does show similar trends for the stiffness, as the standard deviations are relatively small, again indicating a similar trend for the different trials. These deviations are, however, respectively larger than for the trajectory.

Based on these results, we conclude that while the mutations are not reproducible, their general trends are. One reason it may be difficult to reproduce a mutation exactly is that it is impossible to control all the factors that influence the mutation. Although running a simulation reduces the variability of the environment significantly, there are still some factors that cannot be accounted for. An example is the phase differences between the control of the expert and novice robot. As discrete control commands are sent to the robot, small differences in the sent control inputs will influence how the combined task is executed and, therefore, how the mutation occurs. This could be accounted for by using one control node to control both robots, however, this would be counterintuitive as such control modality would undermine the ability of the robots to act as independent agents -— a potentially useful, but completely different use case.

The example of reproducibility presented in this section showed small differences for both the trajectory and the stiffness. In Appendix E examples are given where these differences are larger. However, as the default setting proves the main point – that mutation is not reproducible, but the trends are – it was decided to use the default settings in this paper.

IV. Discussion

The purpose of this research was to explore the occurrence of mutation during robot-to-robot skill propagation to gain insight into the generation of potentially beneficial or dangerous skills. We conducted experiments to confirm the suspicions of skill mutation from [28] and thoroughly explored the underlying mechanisms for mutations.

We tested many parameters to examine various possible situations, and mutations occurred in all of them. The mutations found using these different situations were always different from each other. Furthermore, we also identified differences between the trials for which the same settings were used. We assume that the mutations are not reproducible due to the inability to keep all environmental conditions constant, even in simulation. An example is a slight difference in time between launching the control nodes of the expert and novice. As the controlling is done discretely, a different delay between these nodes will influence the movement of the robot and therefore influence the mutations. Fixing the conditions in the real world is more complicated than in simulation. Therefore, we assume that skill propagation will always lead to some form of mutation when using a collaborative task.

The testing of different settings revealed both benefits and risks associated with the mutations. For instance, as a result of the propagation, new skills may be generated that are useful in various settings. An example of this was seen in the default settings (Fig. 5), where the novice robot's trajectory in run 30 had a significantly increased stroke displacement compared to the original run. The increase in stroke displacement could be beneficial when using a longer saw or if the cut object has a smaller width. While this part of the skill looks quite similar to the original, the same cannot be said for the learned stiffness. We found that the emerged stiffness usually had a value equal to its maximum. For the collaborative movement, this was found to be counterintuitive and energy inefficient, as both robots exert high forces in order to achieve the movement. Nevertheless, while this behaviour might not be beneficial for collaborative tasks requiring reciprocal behaviour, it could be useful in collaborative tasks requiring mirrored behaviour [31], or in adapting the task from collaborative to single-agent where the robot must be stiff all the time.

The increase in the stroke displacement also showed benefits in terms of force manipulability. As the range of the movement increased, the point where the saw switches direction, from now on referred to as the switch-over point, was further away from the base of the robot that had to begin pulling. Therefore, the force manipulability of the robot at this switch-over point was much higher, producing more endpoint force in the sawing axis for the same joint torques. This is desired, as, at this point, the inertia that is needed to build up the speed and momentum of the movement, and the static friction forces are the highest. Therefore, this mutation optimised the movement in terms of the force manipulability of both robots. In addition to the stroke displacement being useful for cases where the saw length increases or the object width decreases, this optimised manipulability also showed to be beneficial for cases where tougher objects should be sawed. For these objects, the static friction would be higher; therefore, more force is needed to pull, which can be gained with improved force manipulability. While this improved manipulability is useful when using two similar types of robots, if one of them is stronger than the other, a large manipulability could result in the stronger robot overpowering the weaker one.

In Fig. 11, we saw an example where the stroke offset (average) changed while the stroke displacement stayed the same. The result showed that when only the expert was controlling the movement (stage 1), low force manipulability would be especially problematic at the switch-over point closest to the base of the robot who was pulling. At this point, the manipulability of the robot is the lowest, resulting in the inability to apply enough force to switch from direction immediately. As a result, the stroke offset shifts towards the expert robot and the force manipulability of one robot increases while it decreases for the other. A better force manipulability would result in less overshoot on the trajectory and, therefore, a smaller shift in the stroke offset. This newly gained skill, where the force manipulability of one robot is much higher than the other, might not be beneficial when two robots are of the same strength, as one would overpower the other. However, if one robot is much stronger, this difference is force manipulability can counteract the overpowering of the weaker one. In this case, the weak robot would use the optimised force manipulability, resulting in a reduced manipulability of the strong robot. While this might be less beneficial for the strong robot's ability to pull, the collaboration between the robots is improved, resulting in better task execution. Examples of the cases where the force manipulability is improved for both robots and where it is improved for one of them have been provided in Appendix F.

Besides developing potentially beneficial skills, we also identified potential hazards in the results obtained with the

default settings. An example of a hazard is where unsafe behaviour occurs due to the size of the forces exerted by the expert, which emerged during the later runs. Initially, forces were between $-50\,\mathrm{N}$ to $50\,\mathrm{N}$, however, the last run showed a range of $-200\,\mathrm{N}$ to $200\,\mathrm{N}$. According to ISO/TS15066 (2016) [43], safety regulations mandate that the force exerted by the robot during human-robot collaboration should not exceed $140\,\mathrm{N}$. Although this might not be critical during robot-robot collaboration (our scenario), it is important in other scenarios involving human-robot collaboration or even when robots work in close proximity to humans. Thus, a force of $200\,\mathrm{N}$ is too high. To account for these high forces, torque limitations have been embedded into the control system. As these limitations are set at the joint level, the resulting behaviour in Cartesian space might be undesired and unpredictable, which is illustrated in Fig. 12. To avoid this, we advise future research to saturate the force at the Cartesian level instead.

We also observed cases where the alternation of the characteristics of the skill were much larger. An example of this is provided for the case where we used a long period (Fig. 10). The results of this setting showed a higher complexity. While less smooth than the original, this behaviour was still suitable for sawing. The resulting behaviour came with the benefit of being more energy efficient as the work exerted by the robot was less. This trade-off between efficiency and smoothness is clear; When the robot can follow the desired trajectory closely, instabilities in the behaviour cause it to be less smooth. At the same time, it requires less effort from the robot to control the movement, making it more energy efficient. The later runs, using the same setting, showed that the trajectories became more complex, which made them less suitable for the sawing task. However, this skill may be beneficial for other tasks that can benefit from complex movements, such as cleaning operations like vacuuming, sweeping, and polishing. Some of these examples can especially benefit from the roughness of the movement; for example, when the robot tries to remove a tough stain, some jerkiness might actually help.

Another example where the behaviour was no longer suitable for the sawing task was presented in Fig. 12. Here, the stroke displacement of the movement was reduced to zero. This skill could be used for a task where the goal is to maintain a specific position, such as holding an object. However, the stiffness of the robot was also reduced to zero, which means that the used stiffness scheme would not be able to maintain the position. An application of a compliant robot in this situation could be to hold an object while allowing another agent to guide the movement. However, these kinds of skills could quite easily be programmed.

To investigate skill mutation, we implemented a sawing task. This task is an excellent case study as it provides many aspects like motion, stiffness, coordination, and physical interaction with both environment and another agent. While the implementation of this task proves the existence of mutations, the influences could be task-specific. To investigate mutation in a more general sense, one should apply the principle of robot skill propagation on other collaborative tasks. With this in mind, future research could separate mutations which are task specific and those which are more general.

In this study, we identified some benefits of mutations, i.e. energy efficiency, optimised force manipulability, and the occurrence of new skills. However, these benefits were mainly examined conceptually and through analysis in this discussion. To gain a better understanding of how mutations affect performance of the task, future research should be conducted to determine if mutations can be beneficial in terms of task performance. If research indicates that task performance decreases, this should not necessarily be seen as a negative outcome. Therefore, further research should also be done to explore if mutated skills could be applied to different conditions/tasks.

After implementing the skill propagation system in the real world, it quickly became evident that executing all experiments would be too time and resource-consuming. Therefore, we decided to conduct experiments using simulation. Using simulation also came with the additional benefit of the ability to more carefully control environmental conditions, which allows for the investigation of how certain external factors affect the mutations. However, simulations may not be able to capture all the complexities of the real world, resulting in disparities between real-world mutations and those observed in the simulations. Despite this, these simulations have still been able to provide valuable insight into understanding skill mutation to a certain degree.

V. Conclusion

This research has taken the initial steps towards filling the knowledge gap concerning mutation that occurs during robot-to-robot skill propagation when executing a collaborative task. Through experimentation, it was observed that mutation occurred in each of the tested conditions. It can thus be inferred that mutation will always happen during skill propagation in collaborative tasks, which answers the first part of the research question. We conducted experiments with the same value for the conditions, and found that the mutations varied, but the trend was similar, which answered the second part of the research question. The variance during each of the conducted trials, even if the settings were kept similar, suggests that mutations are not reproducible, as it is impossible to keep all conditions constant. This therefore answer the third part of the research question.

References

[1] L. Sanneman, C. Fourie, J. A. Shah *et al.*, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *Foundations and Trends® in Robotics*, vol. 8, no. 3, pp. 225–306, 2021.

[2] J. Arents and M. Greitans, "Smart industrial robot control trends, challenges and opportunities within manufacturing," *Applied Sciences*, vol. 12, no. 2, p. 937, 2022.

[3] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," *Springer handbook of robotics*, pp. 1995–2014, 2016.

[4] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial

robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[5] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[7] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[8] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 590–595.

[9] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[10] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[11] P. Kulkarni, J. Kober, R. Babuška, and C. Della Santina, "Learning assembly tasks in a few minutes by combining impedance control and residual recurrent reinforcement learning," *Advanced Intelligent Systems*, vol. 4, no. 1, p. 2100095, 2022.

[12] P. Falco, A. Attawia, M. Saveriano, and D. Lee, "On policy learning robust to irreversible events: An application to robotic in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1482–1489, 2018.

[13] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[14] 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, pp. 297–330, 2020.

[15] A. K. Tanwani and S. Calinon, "A generative model for intention recognition and manipulation assistance in teleoperation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 43–50.

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

[17] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 365–371.

[18] S. Calinon, D. Bruno, M. S. Malekzadeh, T. Nanayakkara, and D. G. Caldwell, "Human–robot skills transfer interfaces for a flexible surgical robot," *Computer methods and programs in biomedicine*, vol. 116, no. 2, pp. 81–96, 2014.

[19] J. Kim, N. Cauli, P. Vicente, B. Damas, F. Cavallo, and J. Santos-Victor, ""icub, clean the table!" a robot learning from demonstration approach using deep neural networks," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 3–9.

[20] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Force-based variable impedance learning for robotic manipulation," *Robotics and Autonomous Systems*, vol. 109, pp. 156–167, 2018.

[21] B. Nemec, F. J. Abu-Dakka, B. Ridge, A. Ude, J. A. Jørgensen, T. R. Savarimuthu, J. Jouffroy, H. G. Petersen, and N. Krüger, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 2013, pp. 1–7.

[22] L. Peternel, T. Petrič, and J. Babič, "Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation," *Autonomous Robots*, vol. 42, no. 1, pp. 1–17, 2018.

[23] M. J. 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.

[24] D. Chen, J. He, G. Chen, X. Yu, M. He, Y. Yang, J. Li, and X. Zhou, "Human-robot skill transfer systems for mobile robot based on multi sensor fusion," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1354–1359.

[25] J. Li, J. Wang, S. Wang, and C. Yang, "Human–robot skill transmission for mobile robot via learning by demonstration," *Neural Computing and Applications*, pp. 1–11, 2021.

[26] V. Chu, T. Fitzgerald, and A. L. Thomaz, "Learning object affordances by leveraging the combination of human-guidance and self-exploration," in *2016 11th ACM/IEEE international conference on human-robot interaction (HRI)*. IEEE, 2016, pp. 221–228.

[27] C. Celemin, G. Maeda, J. Ruiz-del Solar, J. Peters, and J. Kober, "Reinforcement learning of motor skills using policy search and human corrective advice," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1560–1580, 2019.

[28] L. Peternel and A. Ajoudani, "Robots learning from robots: A proof of concept study for co-manipulation tasks," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 484–490.

[29] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[30] 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). IEEE, 2016, pp. 3900–3906.

[31] L. Peternel, N. Tsagarakis, and A. Ajoudani, "A human–robot co-manipulation approach based on human sensorimotor information," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 7, pp. 811–822, 2017.

[32] C. Schindlbeck and S. Haddadin, "Unified passivity-based cartesian force/impedance control for rigid and flexible joint robots via task-energy tanks," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 440–447.

[33] 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, 2003, pp. 3704–3709.

[34] L. Peternel, L. Rozo, D. Caldwell, and A. Ajoudani, "A method for derivation of robot task-frame control authority from repeated sensory observations," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 719–726, 2017.

[35] X. Chen, N. Wang, H. Cheng, and C. Yang, "Neural learning enhanced variable admittance control for human–robot collaboration," *Ieee Access*, vol. 8, pp. 25 727–25 737, 2020.

[36] E. Zheng, Y. Li, Z. Zhao, Q. Wang, and H. Qiao, "An electrical impedance tomography based interface for human–robot collaboration," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 5, pp. 2373–2384, 2020.

[37] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *Proceedings of the ieee/rsj int. conference on intelligent robots and systems (iros2002)*, no. CONF, 2002, pp. 958–963.

[38] 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.

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

[40] L. Peternel, T. Noda, T. Petrič, A. Ude, J. Morimoto, and J. Babič, "Adaptive control of exoskeleton robots for periodic assistive behaviours based on emg feedback minimisation," *PloS one*, vol. 11, no. 2, p. e0148942, 2016.

[41] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[42] T. Petrič, L. Peternel, J. Morimoto, and J. Babič, "Assistive arm-exoskeleton control based on human muscular manipulability," *Frontiers in neurorobotics*, vol. 13, p. 30, 2019.

[43] "Iso/ts 15066:2016—robots and robotic devices—collaborative robots," International Organization for Standardization, Geneva, Switzerland, Tech. Rep., 2016.

[44] A. Gams, A. J. Ijspeert, S. Schaal, and J. Lenarčič, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Autonomous robots*, vol. 27, no. 1, pp. 3–23, 2009.

APPENDIX A
VALIDATION STIFFNESS SCHEME

A. Reference Stiffness Scheme

A continuous stiffness scheme defines the desired stiffness in stage 2 of the learning scheme (Section II-C). Previous work uses a discrete scheme as a reference instead [28]. As the resulting stiffness scheme is continuous, a discrete scheme cannot control the output of the encoding method. We decided to compare the results of both methods with each other to investigate to what extent choosing a different scheme affects the mutation.

A quantitative analysis is provided in Fig. A.1. It shows the evolution of the skill using the continuous and discrete reference stiffness. This result shows that the difference between the learned trajectories is minimal, which can be confirmed by looking at the trajectory analysis in Fig. A.2. This difference is a result of the mutation of the trajectory being independent of the learned stiffness scheme.

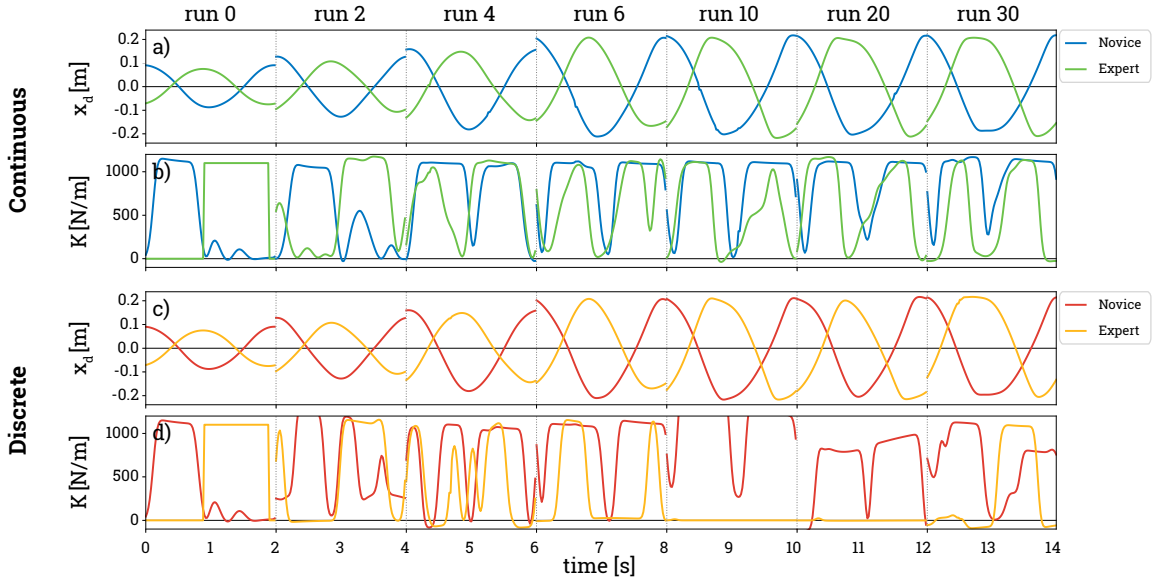


Fig. A.1. Quantitative comparison of the different stiffness schemes. The top two graphs show the evolution of the expert and novice robot's skill (desired trajectory x and stiffness K) using the continuous reference scheme. The bottom two graphs show the evolution of the skill of the expert and novice robots using the discrete reference scheme.

While the trajectory is independent of the used reference stiffness, this is not the same for the learned stiffness. The results in Fig. A.1 clearly show that the learned stiffness scheme is quite different. However, we also saw this for the different trials of the continuous scheme (Section III-E). The interesting parts are observed when looking at the stiffness of the novice and the expert robot. For the continuous scheme, the average stiffness is quite high for both robots. However, for the discrete variant,

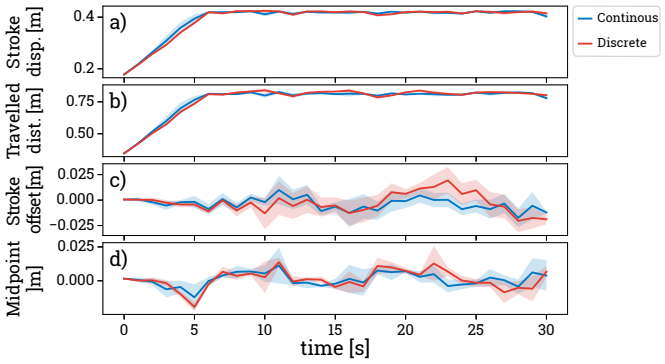


Fig. A.2. Analysis of the learned trajectory of the novice robot during stage 1 of the learning scheme. The stroke displacement, travelled distance, stroke offset, and the midpoint of movement throughout the different runs are shown from top to bottom. Both the results using the continuous and discrete stiffness schemes are shown.

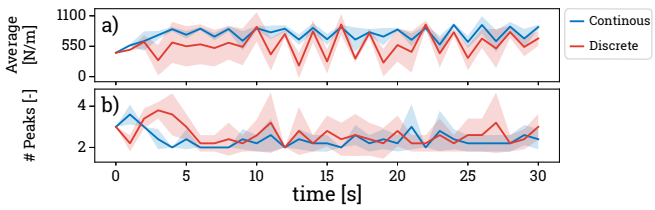


Fig. A.3. Analysis of the learned stiffness scheme of the novice robot during stage 2 of the learning scheme. The average value and the number of peaks per period are shown throughout the different runs from top to bottom. Both the results using the continuous and discrete stiffness schemes are shown.

it is often the case that either the novice has a high value or the expert does (see runs 10 and 20). We found that the runs during which the expert robot has high stiffness and during which the novice robot does, are not necessarily the same for the different trials. Therefore, the standard deviation in the analysis of the learned stiffness (Fig. A.3) is much higher than for the discrete.

The chosen reference stiffness does not affect the learned trajectory. In addition, both methods do not maintain the leader/follower behaviour. As the continuous scheme's encoding can better control the output, we decided to use this scheme.

B. Safety Margin Dynamic Motion Primitives

During the experiment, we encountered an increase in the average value of the stiffness scheme when shifting from stage 2 to stage 3. We identified this as a limitation of the learning method LWR. In Fig. A.4, three different runs of one trial are shown. These results show where no safety regulation was applied when shifting from stage 2 to 3. Run 16 is an example of how the shifting between stages should be. First, the stiffness scheme is learned in stage 2. Then, after around $5\,\mathrm{s}$, the encoding method has captured the main characteristics of the stiffness scheme, which is updated in the final $5\,\mathrm{s}$ of this stage. Finally, when the stage is shifted to stage 3, the stiffness scheme is similar to the last $5\,\mathrm{s}$ of stage 2.

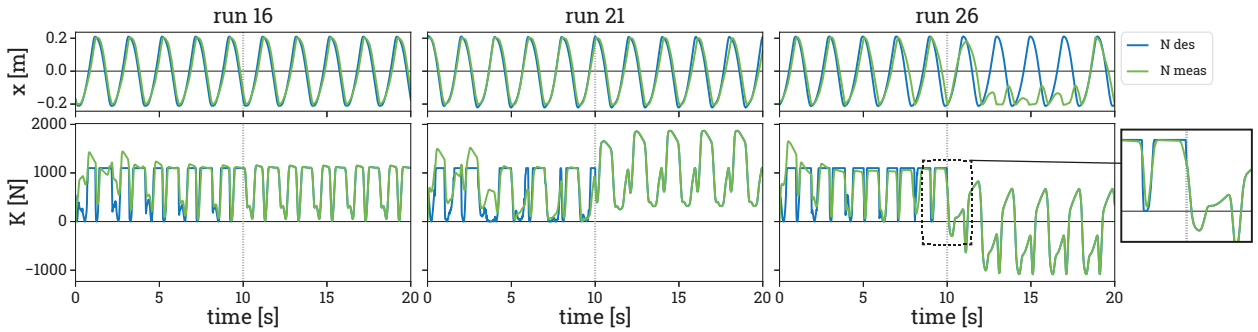


Fig. A.4. Three different runs of a setup using no safety regulations on shifting between stages 2 and 3. For each of the runs, the trajectory and stiffness of stage 2 (0 s to 10 s) and 3 (10 s to 20 s) are shown. Run 16 shows an example of the desired result. Run 21 is an example of a shift between stages resulting in an increased stiffness scheme. Run 26 is an example where the shift between stages results in a decreased stiffness scheme.

We identified two possible bad results of no implementation of safety regulations. An example of the first is shown in run 21. We see that after $5\,\mathrm{s}$, the encoding method can capture the stiffness scheme's characteristics. However, once it shifts to the next stage, the entire stiffness scheme moves up, resulting in a higher average stiffness than learned. While this is not desired, we do not see this badly affecting the measured trajectory.

An example where the measured trajectory is badly affected is in run 26. In this example, the stiffness scheme moves down after shifting to the third stage, resulting in the stiffness often being negative. This shift largely influences the measured trajectory as the robot tries to push itself away from the desired trajectory.

We identified that the shifted stiffness scheme after switching from stage 2 to stage 3 is caused by a large difference between the stiffness value in the last few steps before switching. Therefore, we proposed to implement the following safety regulation

$$\textit{Allowing shifting between stage 2 and 3} = \begin{cases} True & \text{if } |K_t - K_{t-1}| < K_{th} \text{ is true for the last } n_t \text{ time steps,} \\ False & \text{else,} \end{cases} \tag{20}$$

where $K_{th} = 15$ is a threshold, defining the maximum difference between the stiffnesses. The value for this threshold was identified by using a trial-and-error approach. By doing so, we found that if the value was too high, it would be impossible to shift from stage 2 to stage 3. If this value was too low, this method did not account for the shifting of the average as shown in Fig. A.4. A similar effect was visible when tuning the number of time steps n_t, resulting in the choice to investigate the last 5.

APPENDIX B
VALIDATION DYNAMIC MOVEMENT PRIMITIVE AND LOCALLY WEIGHTED REGRESSION PARAMETERS

We investigated multiple studies using DMP and LWR and found that they often use the same values for the parameters [28, 37, 44]. To validate these values, we conducted an experiment in which we changed each parameter individually. For each test, all parameters were set to the default values used in literature besides the one being tested. The literature values are presented in Table B.1. The analysed parameters are the Gaussian kernel N, Gaussian width h, DMP gain α and β, and the forgetting factor λ. As we are tuning the parameters individually, the results might not be optimal. However, we conducted this experiment to validate the values found in literature. A caveat of this approach is that if multiple parameter values are better than the one used in literature, their combination could lead to undesirable results. If this is the case, they should be tested together. However, as the values in Table B.1 show, we only used one value different from literature.

TABLE B.1
VALUES FOR DMP FOUND IN LITERATURE [31, 37, 44], AND THOSE CHOSEN FOR THE EXPERIMENTS.

	N	h	α	β	λ
Literature value	25	2.5N	2	8	0.995
Chosen value	50	2.5N	2	8	0.995

To validate the values of the parameters, we ran a $40\,\mathrm{s}$ test run using the simulation setup (Fig. B.5). The first $20\,\mathrm{s}$ of this run represent stage 1 (Section II-C). This means that the novice robot is compliant, and the expert robot has high stiffness. The first $10\,\mathrm{s}$ of this part will be used to train the DMP of the trajectory by using the measured trajectory of the novice as an input. The second $10\,\mathrm{s}$ is used to validate the resulting DMP. The second $20\,\mathrm{s}$ of the test run are similar to stage 2, in which the novice robot computes a reference stiffness based on the error between the measured and actual trajectory. This stage is used to train and validate the DMP of the stiffness. Again, the first $10\,\mathrm{s}$ are used to learn the DMP and the second $10\,\mathrm{s}$ to validate it.

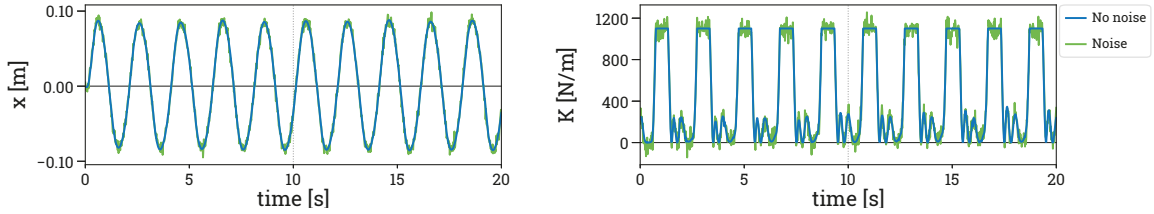


Fig. B.5. Reference signal used for the validation experiment of the DMP and LWR. The left graph shows the trajectory, and the right graph shows the stiffness. The signal without (in blue) and with (in green) noise have been presented for both cases.

To test the robustness of the different values of the parameters, we also conducted an experiment in which random noise was applied to the reference signals. To do so, we added normally distributed noise $p(t)$ to the signal $y(t)$

$$y_r(t) = y(t) + p(t), \tag{21}$$

$$p(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(t-\mu)^2}{2\sigma^2}, \tag{22}$$

where $y_r(t)$ is the resulting signal after adding noise, μ is the mean of the noise set to 0, and σ the standard deviation defined as 5% of the maximum value of the signal y. The result of the reference trajectory with noise and stiffness with noise are also presented in Fig. B.5.

We used three metrics to validate the result of the learned signal; The computational time needed to update the trajectory weights and stiffness weights during one updating step. The mean absolute error (MAE) between the reference and the learned trajectory. And the MAE between the reference and the learned stiffness. The result only showed deviations in the computational time for the experiment validating the amount of Gaussian kernels N. Therefore, we only used this metric for this parameter. In addition to the described metrics, we analysed the data quantitatively by visualising the raw data.

A. Gaussian Kernels

We have determined the value of Gaussian kernels N based on the results presented in Fig. B.6. Fig. B.6b illustrates that computation time is linearly correlated with N, while the error between the reference signal and the computed signal is asymptotic for both the trajectory and stiffness. When no noise is applied to the reference signal, the results in Fig. B.6 demonstrate that no further improvement in MAE is gained when $N \geq 15$ for the trajectory, and $N \geq 25$ for the stiffness. A higher number of kernels means that a higher complexity can be captured, which explains the higher number of kernels required for the stiffness, which shows higher complexity than the trajectory.

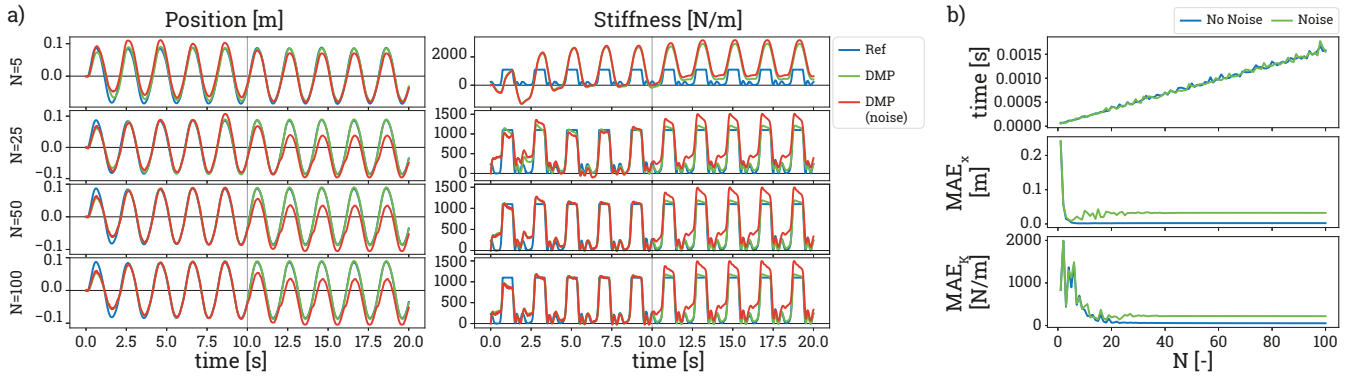


Fig. B.6. The effect of changing the Gaussian kernels N used to learn the DMP using LWR. a) Shows the results of learning the trajectory (left) and stiffness scheme (right) using different values of N. For each of the four cases, the reference signal (blue), the learned DMP using a reference signal without noise, and the learned DMP using a reference signal with noise is shown. b) Show the analysis of the different values for N. The top graph shows the computational time needed to update the weights of the DMP of the trajectory and stiffness. The middle and bottom graphs show the MAE the trajectory and stiffness, respectively.

The trajectory result shows that a low amount of kernels results in the lowest error for the trajectory. This low error results from the reference signal's low complexity, which is confirmed by the stiffness results as the error is quite large. For the trajectory, we see that when switching between the learning and the evolution stage, the average of the learned trajectory scheme moves down. This can be accounted to the inability of the DMP to deal with rapid changes in the reference signal just before switching between these stages (explained in Appendix A-B).

For the stiffness with noise, we see that the DMP can filter out most of the noise, resulting in a similar stiffness as in the case of learning for a signal without noise. For the examples in Fig. B.6a, we see that the computed signal during the validation stage is slightly different from what is visible in the learning stage. We expect this to also result from the switching of stages at a bad time.

Based on the results, we decided to use 50 kernels. The reason for this is that we want the DMP to be able to capture the complexity of the signal. As some later runs will have more complex trajectories and stiffness schemes, we still want the DMP to be able to capture the characteristics. The results also show that for the reference signal with noise, the large number of kernels did not affect the smoothness of the learned signal. Therefore, we concluded that for large values, the DMP could smooth out the noise. That we decided to use a value of 50 kernels instead of 25 comes at the cost of computational time. During experiments, we use time steps 0f $0.2\,\mathrm{s}$. Therefore, a value of $0.0075\,\mathrm{s}$ will not be an issue.

B. Gaussian Width

In [44], the Gaussian width h was set as a function of the number of kernels, i.e., $h = 2.5N$. We decided to change the factor 2.5, and investigate how this influences the ability to follow the reference signal. We define this factor as the Gaussian width coefficient c_h. The result of testing different coefficients has been presented in Fig. B.7.

The results in Fig. B.7b show that for the reference without noise, no improvement is gained for the MAE of the trajectory when $c_h \geq 0.5$. When there is noise, we see in Fig. B.7a that when $c_h \leq 2.5$, the learned trajectory captures the characteristics of the reference signal, but the amplitude and average value are different. When the factor increases, the signal cannot capture these characteristics. These findings have been confirmed by looking at Fig. B.7b.

The MAE of the stiffness, without noise, shows the lowest error when $c_h = 2.5$. When there is noise, this value becomes a bit lower, but $c_h = 2.5$ still seems to be a reasonable choice.

We concluded that $c_h = 2.5$ was the best choice based on the findings. The main reason is that this factor resulted in the lowest error overall for a reference signal without noise. When there was noise, the value of $c_h = 2.5$ fell right between the trajectory's best value and the stiffness' best value.

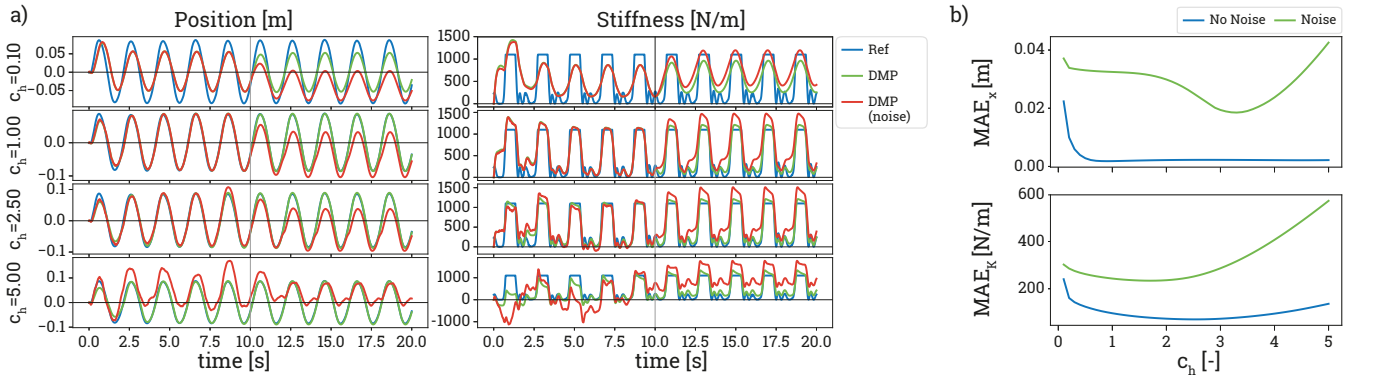


Fig. B.7. Effect of changing the Gaussian width coefficient c_h. a) Shows the results of learning the trajectory (left) and stiffness scheme (right) using different values of c_h. For each of the four cases, the reference signal (blue), the learned DMP using a reference signal without noise, and the learned DMP using a reference signal with noise is shown. b) Shows the MAE between the reference signal with and without noise and the learned signal. The top graph shows the MAE for the trajectory and the bottom graph the one fo MAE for the stiffness.

C. Dynamic Movement Primitive Gains

In Fig. B.8, we show the results of changing the DMP gains α and β. The tuning of β is based on literature [38, 39, 44], which states that when ensuring a relation of 4:1 between α and β, the system would be critically damped. The results of the MAE in Fig. B.8 show an exponentially decreasing for both the trajectory and stiffness. In the case of no noise, when $\alpha \geq 8$ there is no improvement, which occurred around $\alpha \geq 12$ when the reference signal has noise. A downside of using a high value is that the reference trajectory is over-controlled. Therefore, we decided to use a value of $\alpha = 8$ and $\beta = 2$.

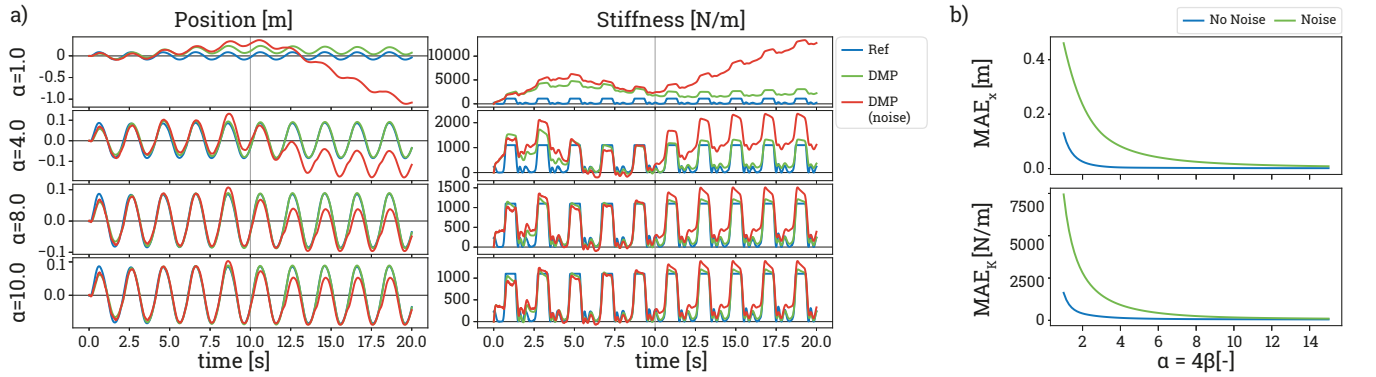


Fig. B.8. Effect of changing the DMP gains α and β, where the system is critically damped: $\beta = \alpha/4$. a) Shows the results of learning the trajectory (left) and stiffness scheme (right) using different values of α. For each of the four cases, the reference signal (blue), the learned DMP using a reference signal without noise, and the learned DMP using a reference signal with noise is shown. b) Shows the MAE between the reference signal without noise and the computed signal. The top and bottom graphs show the MAE of the learned trajectory with and without noise, and stiffness with and without noise, respectively.

D. Forgetting Factor

The results of tuning the forgetting factor $\lambda \in [0, 1]$ are shown in Fig. B.9. This factor defines how much of the previous data is taken into account when computing the DMP [44]. When it is set to a high value, the system does not forget any input value and learns the average of the periodic signal over multiple periods. Moreover, when set low to a low value, it forgets all previous periods, meaning that the resulting DMP is computed using only the last period.

When there is no noise, the value chosen for the forgetting factors does not influence the learned trajectory. We see in Fig. B.9a that even when $\lambda = 0.5$, the learned DMP can capture the characteristics of the signal. For stiffness, this is not the case. For the case that $\lambda = 0.5$, the learned stiffness is not similar to the reference. The results of the MAE of the stiffness (Fig. B.9b) show that when $\lambda \geq 0.85$, there is no improvement in the learned stiffness scheme.

Looking at results when there is noise, we see that a higher forgetting factor always results in a better MAE. We decided to set the value to less than 1, as during the actual experiments, the robot often needs time to adapt to the reference signal. Therefore, it would be beneficial if the first two periods might be taken less into account than the final ones. Based on this, we decided to use a value of 0.995, as it allows forgetting the first part of the learning stage while maximizing the entire learning stage.

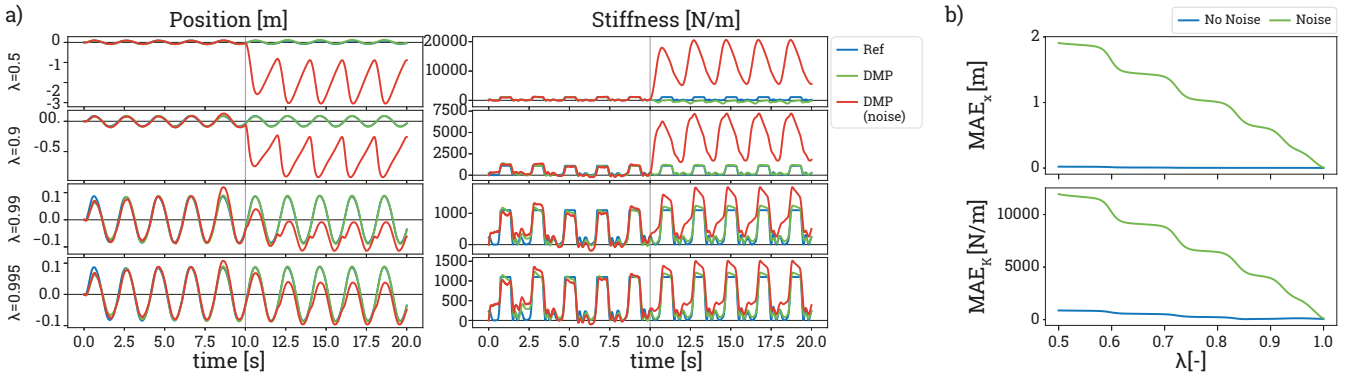


Fig. B.9. Effect of changing the forgetting factor λ used when learning DMP using LWR. a) Shows the results of learning the trajectory (left) and stiffness scheme (right) using different values of λ. For each of the four cases, the reference signal (blue), the learned DMP using a reference signal without noise, and the learned DMP using a reference signal with noise is shown. b) Shows the MAE between the reference signal without noise and the computed signal. The top and bottom graphs show the MAE of the learned trajectory with and without noise, and stiffness with and without noise, respectively.

APPENDIX C
EXPERIMENTAL SETUP

A. Robot Control

To control the robots, we used Robot Operating System (ROS). The ROS framework uses processes (nodes) that enable parts of the control structure to be individually designed. Communication between these different nodes can be done by streaming data over topics. A graphical representation of the implemented control structure is provided in Fig. C.10. The rectangle boxes illustrate the nodes, and the lines connecting the different nodes are the topics.

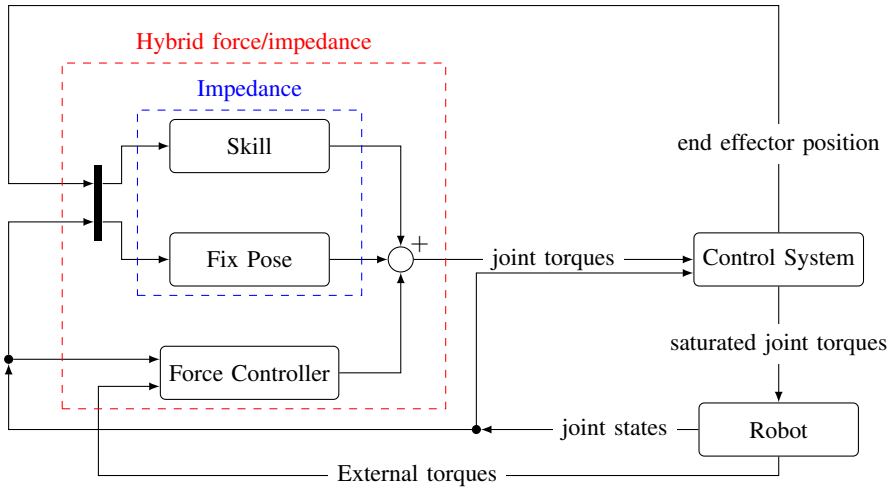


Fig. C.10. The control scheme of the robots, controlled using ROS. The expert and the novice robot are controlled using a hybrid force/impedance controller. The impedance controller comprises two components: *Skill* and *Fix Pose*. *Skill* calculates a force using a desired trajectory and stiffness, both based on the phase ϕ. *Fix Pose*, on the other hand, computes the force using a fixed reference position and a fixed stiffness. Both of these nodes use the measured end effector state/position to compute the impedance force. The *Force Controller* uses the measured external force, determined by translating the external torques to forces using the Jacobian, to compare with the reference force of $5\,\mathrm{N}$. Each force, computed by the different parts of the hybrid controller, is translated to torques using the Jacobian. They are combined and sent to the *Control System*, which, if needed, saturates them based on the defined torque limits. This node will then send the saturated joint torques to the *Robot*.

Both the expert and the novice robot are controlled using a hybrid force/impedance controller ((1)), which has been highlighted in red in Fig. C.10. The implementation in ROS divides this controller into three parts:

- *Skill*: This node computes forces required to control the sawing movement along the x-axis of the concerning robot by means of the impedance law ((4)). The input of the impedance controller (desired trajectory and stiffness scheme) consists of an existing skill for the expert robot and a learning method for the novice robot. For both cases, the control depends on the phase ϕ of the movement, the current stage of the three-staged learning scheme (Section II-C) and the measured position of the end effector of the robot. The desired trajectory and stiffness are determined based on the phase and stage. The impedance force is translated to torques, using the Jacobian computed with the model of the robot and the joint states ((5)). This is required, as the robots are controlled on the joint level and not in the Cartesian space.
- *Fix Pose*: This node also computes forces using the impedance law. However, in contrast to the skill, the reference positions are fixed. This node computes translation forces along the y-axis and rotational forces along each axis. The computed force is translated to torques using the Jacobian.
- *Force controller*: This node implements the force control part of the hybrid controller ((2)), ensuring the saw makes contact with the object. This node receives the external torques acting on the robots, which are translated to end-effector forces using the Jacobian. Based on this force and the reference force of $5.0\,\mathrm{N}$, a resulting force along the z-axis is computed, which is translated to torques by using the Jacobian.

The combined torques, consisting of those computed using an impedance controller and those using the force controller, are sent to the *Control System*. This system determines whether the torques exceed the predefined torque limits. The torques that exceed this limit are saturated. The resulting torques are sent to the *Robot*. In addition, the *Control System* computes the end effector states of the robot, by translating the joint states using the Jacobian. The hybrid impedance/force nodes use the resulting end effector position as an input.

The *Robot* describes the robot in the real world or the robot in simulation (Gazebo). After receiving the torques of the control system, it should move to the desired position while maintaining a certain reference force. This node also sends the current states of the robot, which consist of the joint positions and the external torques acting on the robot.

B. Real World Setup

The real-world setup consists of two robots (iiwa7 and iiwa14), a saw, a clamp, and an object. This setup is visualised in Fig. C.11.

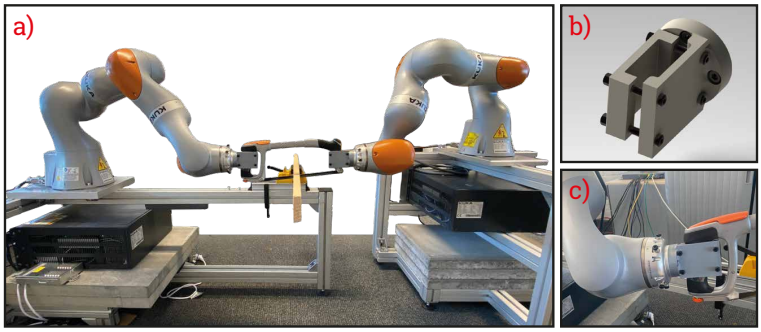


Fig. C.11. Setup used for real world experiments. a) Shows the complete setup including a iiwa7 (right) and iiwa14 (left) robot, the BACHO metal saw, and the wooden plank that is sawn. b) Visualises the SolidWorks render of the clamp used to attach the saw to the end-effector of the robot. c) Presents a close-up of the saw attached to the robot.

We needed to calculate a rotation matrix to ensure the local frame of the robot's end-effector was aligned with the global frame since the perfect alignment of the robots was not possible. To do this, we put the robot's end-effector into its initial position, which was then used as the origin of the local frame. We then moved the robot along the x-axis of the global frame, measuring the positions as we went. Using these data points, we calculated the rotation matrix. In the different stages of the learning scheme, the robot's movements were controlled by rotating the measured positions of the end-effector, computing the corresponding force in the local frame, and then rotating the force back to the base frame of the robot.

C. Simulation Setup

The main principle of the setup used in simulation is the same as in the real world; There are two robots, a saw and an object. The setup is visually represented in Fig. C.12. The main difference with the real-world setup is the saw and how it is attached to the robot. This has been visualized in Fig. C.13. Instead of a saw, we simulated a rectangular box with a shape of L_{saw}x0.05x0.05 m. The length of the saw L_{saw} variates between the different setups. For the default settings, the saw has a length of $0.45\,\mathrm{m}$. The weight of the saw is equally distributed. This weight also variates between the different setups, with a default setting of $0.5\,\mathrm{kg}$.

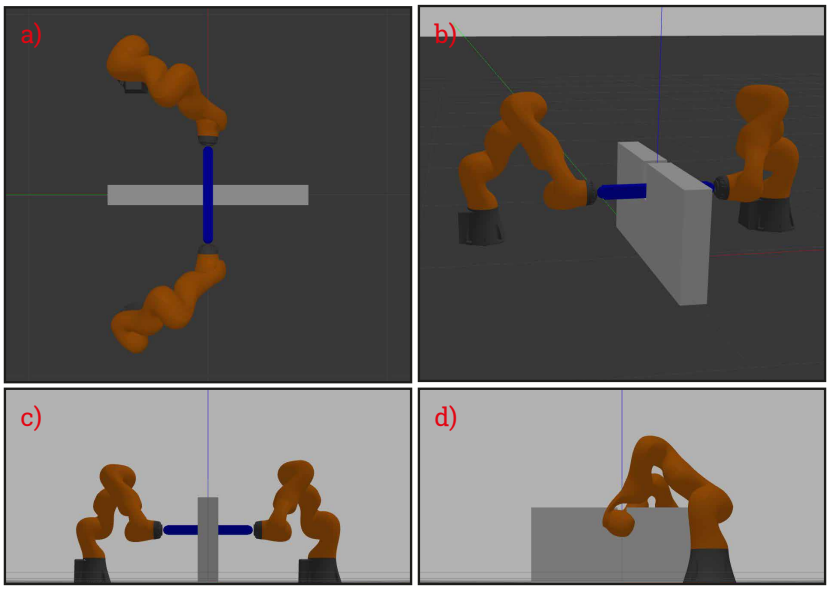


Fig. C.12. The setup used for the simulation experiments. a) - d) Show the setup's top view, 3D-view, front view and side view. These figures show the iiwa7 (left in c) and iiwa14 (right in c), the blue-coloured simulated saw, and a white object.

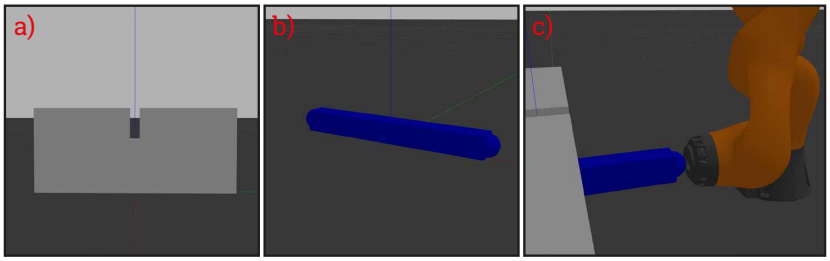


Fig. C.13. Object and saw used in the simulation environment. a) shows the object, which has a grove in which the saw fits. b) Shows the simulated saw. c) Shows the attachment of the saw to the robot's end-effector, which is done using a virtual joint.

APPENDIX D
INFLUENCE OF EXTERNAL FACTORS ON MUTATION

In addition to the external factors presented in Section III-D, we also tested multiple other factors. This appendix will illustrate these factors and give a general analysis of them. The factors tested are as follows: the base position, the friction coefficient, the initial stroke displacement, the length of the saw, the mass of the saw, the maximum stiffness value, the length of the period, the switching of the type (iiwa7 or iiwa14) robot being an expert and the robot being a novice, and the threshold. Some of these factors are visually represented in Fig. D.14. In addition, a comparison of these factors is shown in Table D.2, which is an extended version of Table II.

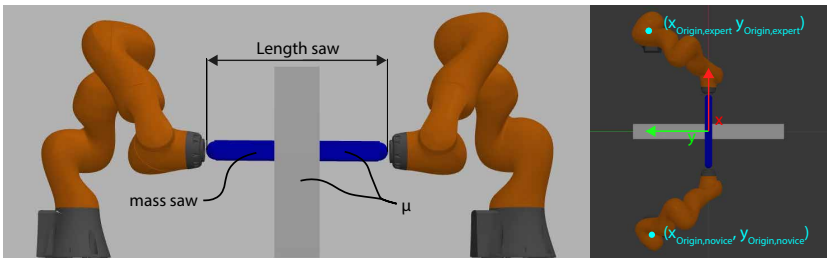


Fig. D.14. Visualisation of the robots in `Gazebo`. The left illustration is a side view of the setup and the right illustration a top view.

TABLE D.2
EXTENDED COMPARISON OF THE EXTERNAL FACTORS.

Factor	Symbol	Default value	Value	Same as default	Repeatable[1]	Converges
Base position	-	Both at y = 0.4m	Both at y=0.0 m	No	Yes	No
			y=0.4 m/y=−0.4 m	Yes	Yes	Yes[2]
			y=0.4 m/y=−0.5 m	Yes	Yes	Yes[2]
Friction coef.	μ	0.0	0.05	No	Yes	Yes
			0.01	No	Yes	Yes[2]
			0.005	No	Yes	Yes[2]
Initial stroke displacement	Δx_0	0.15 m	0.15 m	No	Yes	Yes[2]
			0.2 m	No	Yes	Yes[2]
Length saw	L_{saw}	0.45 m	0.65 m	No	No	Yes[2]
			1.05 m	No	No	Yes
Mass saw	m_{saw}	0.5 kg	3.0 kg	No	Yes	Yes[2]
			10.0 kg	No	No	Yes
Maximum stiffness	K_{max}	1100 N	500 N	No	Yes	Yes[2]
			2200 N	No	Yes	Yes[2]
			5000 N	No	Yes	Yes[2]
Period	τ	2 s	1 s	No	No	Yes[2]
			3 s	No	No	No
Switch Robot	N/A	True	False	No	Yes	Yes[2]
Threshold	e_{th}	0.02	0.01	Yes	Yes	Yes[2]
			0.04	Yes	Yes	Yes[2]

[1] Repeatable is set to "yes" if the main trends of the different trials are similar. As discussed in Section III-E, the mutations are never repeatable throughout the different trials, but often their main trends are.
[2] This mutation converges due to approaching some limit, i.e. an environmental boundary, a joint limit or a torque limit.

The *based position* describes the origin of both robots (iiwa7 and iiwa14). The value given in Table D.2 gives the y coordinate of the origin of both robots, as defined in Fig. D.14. The x coordinate depends on the length of the saw. In case of the default length of $0.45\,\mathrm{m}$, this coordinate has a value of $0.7\,\mathrm{m}$ or $-0.7\,\mathrm{m}$, depending on the type of robot. When this length is changed, the value has increased (or decreased) half the difference in length. So for a length of $0.65\,\mathrm{m}$, the values are set to $0.8\,\mathrm{m}$ or $-0.8\,\mathrm{m}$, and for a length of $1.05\,\mathrm{m}$ to $1.0\,\mathrm{m}$ or $1.0\,\mathrm{m}$. The reason for this is that, in this case, the initial joint positions of the robot have not changed. This way, the influence of increasing the saw length can solely be investigated, as otherwise, the joint configuration would also play a part. The results of this factor have already been discussed in Section III-D.

The results of the period τ (one sequence of movement), the friction coefficient (ratio between friction force and normal force), and the maximum stiffness K_{max}, have also been discussed in Section III-D. In addition, an extensive analysis of the results of the type of stiffness scheme (continuous or discrete) is presented in Appendix A.

The *initial stroke displacement* of the movement is the difference between the maximum and minimum x-position during the first run. Changing this value showed a difference in the amount of run until the boundary of the environment was reached. Increasing the initial stroke displacement seemed to have a linear correlation with the number of runs until the environmental

boundary was reached. As a small initial stroke displacement of $0.1\,\mathrm{m}$ has a smaller initial value than the default of $0.15\,\mathrm{m}$, it would thus take more time to reach this boundary.

The result of changing the *length of the saw* at first glance looked interesting, as the variability in the results was relatively high. This variability would indicate that the mutation was not repeatable. The results also showed not to converge to an environmental boundary or one of the robot limits. However, upon closer inspection, it was found that the reference force of $-5\,\mathrm{N}$ was not large enough to remain in contact between the saw and the object. This resulted in the saw floating in the air. As a result, the robot had more freedom to rotate around, which largely influenced the x position.

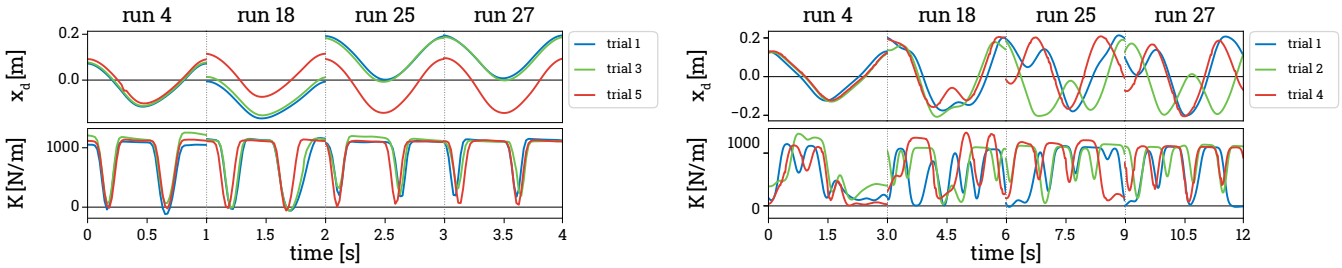
In simulation, the *mass* of the saw has been equally distributed along the saw. Increasing the mass of the saw to $3.0\,\mathrm{kg}$ showed only a small difference compared to the default setting of $0.5\,\mathrm{kg}$. This difference was visible in the number of runs until the boundary of the environment was reached, which was less for a larger weight. In the case where the mass was increased quite significantly ($10.0\,\mathrm{kg}$), it was visible that the stroke displacement stagnated around $0.3\,\mathrm{m}$ instead of $0.4\,\mathrm{m}$. This is because the expert keeps overshooting the desired trajectory on the side closest to itself. A zigzag behaviour occurred as we switched between the type of robot used as the expert between the different runs.

If the *switching of robot* is set to true, the type of robot (iiwa7 and iiwa14) used as the expert is different after each run; For the even number runs, the iiwa14 is the expert, and for the odd number runs iiwa7 is. In case the switching of the robot is set to false, the expert is always the iiwa7. No switching means that the expert robot uses a skill which was learned by another type, namely the iiwa14. The results of not switching showed no zigzag patterns in the analysis. We were expecting the expert robot to averagely move the trajectory closer to itself, as this was visible when the robots switched between being the expert. However, this does not happen. Instead, the results are quite similar to the setup where the robot is switched. The results might have been more interesting if we applied this no-switching approach, to a more complex case, for example, when the period is long.

The threshold value e_{th} was defined in (9). As the trajectory is learned in stage 1, when the stiffness used is of a constant high value or zero, the threshold did not influence the mutation of the trajectory. The results on the stiffness scheme were more significant, as a higher threshold resulted in more complexity in the scheme and a lower average value. In the case of a value of $e_{th} = 0.4$, we observe some follower/leader behaviour when looking at the results. However, this pattern is significantly less clear than what was visible during the initial run.

APPENDIX E
REPRODUCIBILITY OF MUTATIONS

We investigated the repeatability of the skill mutation by comparing the results, using the same settings for the external factors, during different trials. In Section III-E we discussed the reproducibility of the mutation using the default settings. These results showed that the mutation was not repeatable, but the main trends found in the analysis were. In Fig. E.15, two other examples, which are more extreme than the one shown previously, are presented. These cases make use of a different period than the default.



(a) (b)

Fig. E.15. Reproducibility of the mutation. Figures E.15a and E.15b show the reproducibility using a period of 1 s and 3 s respectively. In each subfigure, the top graphs illustrate the learned trajectory, and the bottom graphs illustrate the learned stiffness. Different runs are shown with the results for three trials from left to right.

For a short period (1 s), it is visible that the mutation of the trajectory is not identical for each trial. While the shape of the learned trajectory stays quite similar, the average value during each trial is shifted. In contrast, the stiffness scheme is quite similar for each trial. This is because the reference of the stiffness usually remains at its highest value, except when the desired trajectory intersects the actual trajectory, which occurs twice per period.

The results for a long period (3 s) shown in Fig. E.15b show both differences between the trajectories and the stiffnesses for the different trials. For the trajectory, we see similar mutations throughout the different runs. However, for some trials, specific mutations have been reached during earlier runs than for other trials. For example, trials 1 and 3 have mutated more extensively than trial 5 in run 27.

We do not necessarily see a pattern when observing the stiffness in Fig. E.15b, which results from the trajectory being more complex. The analysis of the different trials is similar (high complexity, same average value). However, this trend does not seem to be visible in the raw data.

APPENDIX F
FORCE MANIPULABILITY

A. Calculation of Force Manipulability

The manipulability of the robot can be computed by means of the $\boldsymbol{J}$ of a robot [42]. This Jacobian describes the relationship between the joint velocities and endpoint (end-effector) velocities of the robots. The relationship between the joint torques and endpoint forces are described using $\boldsymbol{J}^{-T}$. We can define an ellipsoid which maps all possible variables in the joints space to the endpoints in the Cartesian space. As set of these joint torques is described by

$$||\boldsymbol{\tau}||^2 = \boldsymbol{\tau}^T\boldsymbol{\tau} \leq 1, \tag{23}$$

where $\boldsymbol{\tau}$ is the joint torque vector. As the transformation of from joint torques to enforces is given by

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{F}, \tag{24}$$

where $\boldsymbol{F}$ are the end effectors Cartesian forces/torques and $\boldsymbol{q}$ the joint angle vectors, we can define (23) as follows

$$||\boldsymbol{J}^T\boldsymbol{F}||^2 = \boldsymbol{F}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)\boldsymbol{F} \leq 1, \tag{25}$$

where the inner product $(\boldsymbol{J}\boldsymbol{J}^T)^{-1} = \boldsymbol{M}_F$ is used to compute the force manipulability. We can decompose $\boldsymbol{M}_F$ by means of its eigenvalues and eigenvectors

$$\boldsymbol{M}_F = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}, \tag{26}$$

where $\boldsymbol{Q} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3]$ is a matrix containing the eigenvectors, and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1\lambda_2\lambda_3)$ a diagonal matrix with the eigenvalues. Filling in (26) therefore results into

$$\begin{aligned}
\boldsymbol{M}_F &= \begin{pmatrix} v_{1x} & v_{2x} & v_{3x} \\ v_{1y} & v_{2y} & v_{3y} \\ v_{1z} & v_{2z} & v_{3z} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} v_{1x} & v_{1y} & v_{1z} \\ v_{2x} & v_{2y} & v_{2z} \\ v_{3x} & v_{3y} & v_{3z} \end{pmatrix} \\
&= \begin{pmatrix} v_{1x} & v_{2x} & v_{3x} \\ v_{1y} & v_{2y} & v_{3y} \\ v_{1z} & v_{2z} & v_{3z} \end{pmatrix} \begin{pmatrix} \lambda_1 v_{1x} & \lambda_1 v_{1y} & \lambda_1 v_{1z} \\ \lambda_2 v_{2x} & \lambda_2 v_{2y} & \lambda_3 v_{2z} \\ \lambda_3 v_{3x} & \lambda_2 v_{3y} & \lambda_3 v_{3z} \end{pmatrix} \\
&= \begin{pmatrix} \lambda_1 v_{1x}^2 + \lambda_2 v_{2x}^2 + \lambda_3 v_{3x}^2 & \lambda_1 v_{1x}v_{1y} + \lambda_2 v_{2x}v_{2y} + \lambda_3 v_{3x}v_{3y} & \cdots \\ \lambda_1 v_{1y}v_{1x} + \lambda_2 v_{2y}v_{2x} + \lambda_3 v_{3y}v_{3x} & \lambda_1 v_{1y}^2 + \lambda_2 v_{2y}^2 + \lambda_3 v_{3y}^2 & \cdots \\ \lambda_1 v_{1y}v_{1x} + \lambda_2 v_{2y}v_{2x} + \lambda_3 v_{3y}v_{3x} & \lambda_1 v_{1z}v_{1x} + \lambda_2 v_{2z}v_{2x} + \lambda_3 v_{3z}v_{3x} & \cdots \end{pmatrix} \\
&\quad \begin{pmatrix} \lambda_1 v_{1x}v_{1z} + \lambda_2 v_{2x}v_{2z} + \lambda_3 v_{3x}v_{3z} \\ \lambda_1 v_{1y}v_{1z} + \lambda_2 v_{2y}v_{2z} + \lambda_3 v_{3y}v_{3z} \\ \lambda_1 v_{1z}^2 + \lambda_2 v_{2z}^2 + \lambda_3 v_{3z}^2 \end{pmatrix} \\
&= \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix}.
\end{aligned} \tag{27}$$

We are interested in finding the manipulability along the x-axis, as this is the axis of the sawing movement. Therefore, we can define $\boldsymbol{F} = [F_x, 0, 0]^T$. If we rewrite (25) and fill it, we get

$$\begin{aligned}
\begin{pmatrix} F_x & 0 & 0 \end{pmatrix} \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix} \begin{pmatrix} F_x \\ 0 \\ 0 \end{pmatrix} &\leq 1 \\
\begin{pmatrix} m_{1,1}F_x & m_{1,2}F_x & m_{1,3}F_x \end{pmatrix} \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix} &\leq 1 \\
m_{1,1}F_x^2 &\leq 1
\end{aligned} \tag{28}$$

We are interested in the maximum value of the manipulability, therefore, we can rewrite (28) and find the formula of the manipulability along the x-axis as follows

$$F_x = \sqrt{\frac{1}{m_{1,1}}} = \sqrt{\left(\lambda_1 v_{1x}^2 + \lambda_2 v_{2x}^2 + \lambda_3 v_{3x}^2\right)^{-1}} \tag{29}$$

B. Case Studies

In the results, we found two different case studies for force manipulability. In case 1, the stroke displacement of the robot increases. This results in both robots optimising their force manipulability when they first need to pull. For case 2, one robot has optimised its force manipulability while the other becomes slightly worse. Figures F.16 and F.17 show stage 3 of two runs (run 2 and run 30). For both cases, we see in the upper graph the trajectory of the expert and novice robot and the bottom its force manipulability calculated using (29). These trajectories have been presented in the global frame, where the base of the iiwa7 and iiwa14 are located at $x = -0.7\,\mathrm{m}$ and $x = 0.7\,\mathrm{m}$ respectively.

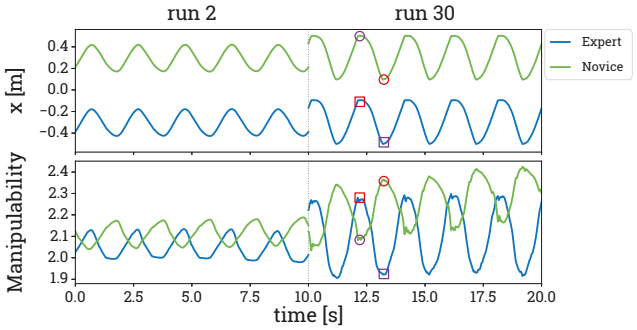


Fig. F.16. Force manipulability of both the expert and novice robots during two runs, obtained when their base is displaced from the x-axis with $0.4\,\mathrm{m}$.

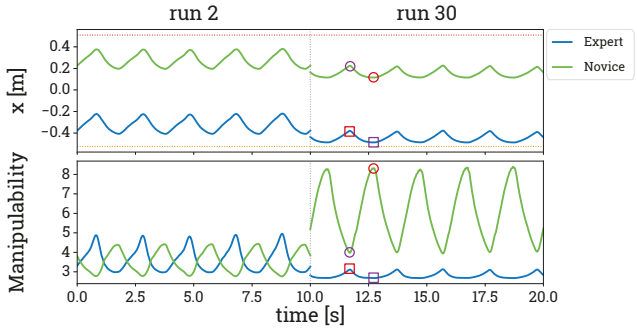


Fig. F.17. Force manipulability of both the expert and novice robots during two runs, obtained when their base is aligned with the x-axis.

For both cases, we have identified two points of the expert (with squares) and two points of the novice (with circles). The red-coloured markers indicate the point when the end-effector is furthest away of its base position. After this point, the movement will, for this specific robot, switch from a pushing to a pulling movement. At this point, the force manipulability of the robot is always the highest, as the end-effector is the furthest away from its base position. On the other hand, the purple-coloured markers indicate the point when the end-effector is the closest to the base of the robot, meaning after this point, they will switch from a pulling to a pushing movement. At this point, the force manipulability is always the lowest, as it is closest to its base position.

The results of case 1 (Fig. F.16) were obtained by using the default settings, where both robots were located at a y position of $0.4\,\mathrm{m}$. In Section III-C, we already discussed that an increase in the stroke displacement was visible throughout the results. The results show that this increase in stroke displacement, which is visible when comparing run 2 with run 30, also influences the force manipulability of the robots. As the stroke displacement is almost double, so does the amplitude of both the force manipulability. While the range of movement of both robots is averagely just as far from their base position, we see that the resulting force manipulability is not the same. This means that one of these robots (the novice)is stronger than the other one.

That one of the robots is stronger than the other, has especially been highlighted in Fig. F.17. These results were obtained when both robots were aligned with the x-axis, meaning $y = 0.0\,\mathrm{m}$. This graph also shows two dotted lines, which indicate the limits of the movement of both robots along the x-axis, which result from the joint limits; The limit of the expert is indicated by the orange line, and the ones of the novice by the red one. We found these values by moving the end-effector of the robot as close as possible to the base of the robot, while maintaining a fixed reference position of $y = 0.0\,\mathrm{m}$ and z=$0.3\,\mathrm{m}$. For the presented cases, We can see that the entire movement moves away from the novice by comparing run 2 to run 30. As a result, the force manipulability of this robot significantly increases.In contrast, the force manipulability of the expert robot decreases. However, the decrease in force manipulability of the expert is significantly smaller than the increase of the novice. The results indicate that the novice robot was stronger than the expert one. While this behaviour is thus not desired for this setting, as the novice robot will overpower the expert during movement, it can be useful when we would switch the learned skill between the robots. In this case, the novice robot would be stronger, while the expert gains the benefit of being in a better configuration, which allows for higher force manipulability. Therefore, combined, they would execute the task much better.