Legible Grasping with Tele-operated Learning from Demonstration

Thesis Report

M.H. van Beem

Delft University of Technology

TU Delft

Legible Grasping with Teleoperated Learning from Demonstration

by

M.H. van Beem

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday June 17, 2022 at 14:00.

Student Number:	4354206	
Thesis Committee	Prof. dr. R. Babuska	TU Delft, supervisor
	Ir. D. Karageorgos,	Heemskerk Innovative Technology, supervisor
	Dr. J. Kober,	TU Delft
Institution:	Delft University of Technology	

TU Delft

Preface

In order to relieve the health care sector, Heemskerk Innovative Technology (HIT)[1] is developing a Remotely Operated Service Robot (ROSE). The robot will reduce the workload of care workers by taking over Activities of Daily Living (ADLs) within care homes. Picking up objects, cleaning, and clearing a table are examples of ADLs. To be efficient, the robot should perform such tasks as autonomously as possible. This can be accomplished by planning safe motions. Although robots move in a safe manner, humans can still be afraid of the way they move in their environment. My role at Heemskerk Innovative Technology was to develop grasp movements that are perceived as legible, so that the robot can be accepted as a daily assistant.

A suitable example of legibility being required in an ADL task would be a human holding a cup and a robot pouring a liquid into the cup. In order to accomplish this, the robot should pick up a bottle, move towards the cup, pour a drink into the cup, and place the bottle back. In general, many other ADLs involve a pick-and-place procedure. Motion planners are generally used in order to execute pick-and-place tasks. However, after implementing a motion planner we saw that the motions of the robot are non-intuitive. The presence of a table in the environment causes the robot's movements to become even less predictable. The planner uses a completely known environment to generate a collision-free motion that approaches the goal pose. It is possible to add social constraints to the motion planner in order to make it more legible and make the human feel more comfortable. However, humans can still be afraid when an unclear motion, that does not show intent, is executed. Therefore, there are other ways to make the intention clear through optimizations or reinforcement learning. Nevertheless, reinforcement learning is not preferred as it is based on trial-and-error. These optimization methods have limitations when applied to various situations. In this thesis, we will try to make legible grasps by using Learning from Demonstration (LfD). The robot is equipped with a haptic device that facilitates remote control. For that reason, a teleoperated Learning from Demonstration module has been developed to create legible grasps, and it has been tested both in simulation and on a real TIAGo robot.

I want to thank my supervisor from the TU Delft, Robert Babuska for the confidence he gave me, as well as his feedback. Furthermore I would like to thank my supervisor Dimitris Karageorgos and all others at HIT. They gave me valuable feedback, input and energy to finish my work. Having the Covid restrictions at the beginning of the thesis prevented me from getting to know the company well. Nevertheless, I enjoyed my time at HIT at the end of my thesis. It was an excellent learning experience and I am grateful to have had the opportunity to graduate at a company. I wish to express my sincerest gratitude to all those who have supported me throughout the years. Specifically, my parents and girlfriend, who have always been there for me.

M.H. van Beem
Delft, June 2022

[1] https://heemskerk-innovative.nl/

Contents

1

Paper

Legible Grasping with Teleoperated Learning from Demonstration

Marnix van Beem, Robert Babuška, Dimitris Karageorgos

***Abstract*— State-of-the-art object grasping with 7-DOF robotic manipulators requires joint configuration planning methods in order to provide position control of the end-effector. These motion planners are able to calculate a motion plan to execute a safe grasp, while taking environmental constraints into account. In human-robot interaction, a well known problem is that humans are uneasy with the arm motion the robot executes, because the motion plan lacks parametrization of variables which would account for the impression of legibility. In this study we develop a method which allows for teleoperated learning from a single demonstration that is perceived more legible by humans. The operator uses the Geomagic Touch haptic device to demonstrate a movement of the robot's end-effector. Modeling a motion path from a single teleoperated demonstration is achieved using Dynamic Movement Primitives. The effectiveness of the teleoperated LfD module has been demonstrated both in simulation and on a TIAGo robot in a variety of poses. An experiment is conducted in which a state-of-the-art motion planner was compared to the proposed LfD method and the ability of human participants to predict the goal object of the robot. Using the teleoperated LfD method, the ability to predict the goal objects increases significantly and the human is more confident in making the prediction (P = 0.0102 and $P < 0.001$, respectively). This means that with the learning method a more legible grasp was generated than with the state-of-the-art motion planner.**

I. Introduction

In order to relieve the health care sector, Heemskerk Innovative Technology (HiT)[1] is developing a Remotely Operated Service Robot (ROSE). It is intended to reduce the workload of healthcare workers which is partially achieved by performing Activities of Daily Living (ADLs). For instance, picking up objects, cleaning, and clearing a table are examples of ADLs. The robot should be able to carry out these activities as autonomously as possible. This allows healthcare workers to devote more time to patient care as a result of the robot performing these tasks. It is necessary for the robot to be able to manipulate (and grasp) objects in order to perform the above-mentioned tasks. The focus of this thesis is to automate grasping of objects, by remotely teaching the robot on how to perform the grasp in a human-friendly way.

For several decades, robots have been used in industrial environments to perform a variety of tasks. Often robots are physically separated from humans to ensure safety. However, more recent research has focused on collaboration between humans and robots. This requires a robot to be safe, reliable, and legible during operations for a human to accept it as an assistant [1]. The term "legible" may also refer to something that is readable, understandable, or intent-expressive. Dragan et al.'s [2] definition of legibility is used throughout this paper. A legible motion is the one that enables the observer (human) to confidently and quickly predict the robot's intention. It is possible for the robot to use complementary motions in order to increase legibility. For example, looking at the object during a hand-over motion or pointing to an object [3], [4], [5]. However, the end-effector motion for a manipulation task can also be utilized to demonstrate legibility. Consequently, the motion fulfills the required task and enables the observer to predict the robot's behavior. A more legible robot motion offers humans a greater sense of safety and comfort [1]. Furthermore, when the robot's movement indicates its intent, it will be more acceptable as a personal assistant [6].

The first author is with the Cognitive Robotics department, Delft University of Technology, Mekelweg 2, 2628 CD Delft, Netherlands.

[1] https://heemskerk-innovative.nl/

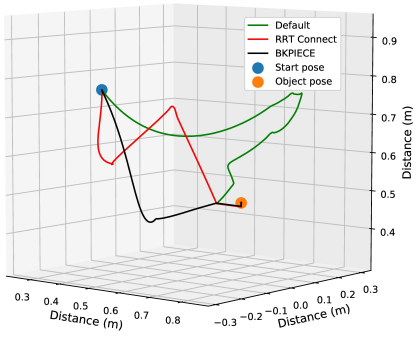


Fig. 1: Grasp motion generated by three different motion planners. The start pose of the end-effector (blue dot) and the object pose (orange dot) are for all planners the same. The default MoveIt motion planner (green path) is the Lazy Bi-directional KPIECE planner. The intention of the robot is probably not that clear for the human in this situation.

Motion planners are commonly used to produce robot manipulation tasks. A motion planner takes into account a path plan as well as what the robot needs to do to complete that path. However, the intention of the motion is not considered in this process. Resulting in motions that is non-intuitive for humans. Figure 1 shows three motions of the end-effector generated by motion planners using MoveIt. However, the robot does not provide any indication of what it will do. There is unpredictability in the movement of the end-effector and joints (which are not visible in the illustration). To solve this problem, humans could become familiar with the movements of robots. This solution, however, is more effective when the robot moves naturally [7]. In addition, when using not an optimal planner the robot does not always plan the exact same movements. Even in a similar situation and using the same planner, the robot will still have a variety of ways to grasp the object. Three different end-effector movements are shown in Figure 2 which are generated using the same planner and in the exact same situation. It is difficult for humans to comprehend why the robot moves in such a strange manner.

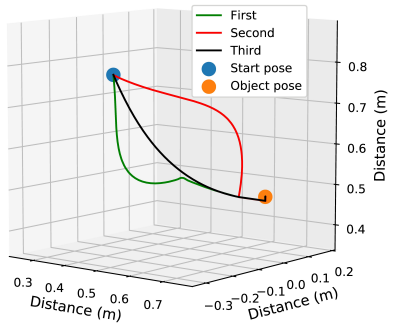


Fig. 2: Result of planning three times a motion for the end-effector in the same situation (same start pose of the robot, same environment, and same object pose) and with the same planner. All three motions are different, which makes it unpredictable what the robot will do.

Another aspect of legible grasping is that the object should be held in a way that mimics human behavior. An object's shape plays a role in grasping it in a human-like manner. When handling a cereal box, for example, the narrow side should be grasped, not the wide side. Another example is that the robot could pick up a coffee mug using its handle. Illustration 3 shows how the robot could grasp a cereal box. Thus, the robot should indicate which object it will grasp as well as where it will grasp it.

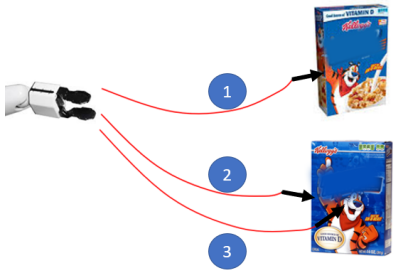


Fig. 3: An example of grasping a cereal box. The preferred method is to grab the box by the narrow side. In the figure, motion 1 is depicted as the preferred grasp. The robot should understand that motion 2 will be executed if the box is placed elsewhere.

Hence, there is a need for robots with knowledge of legible grasping. As humans, we learn to infer intentions from other humans' motions [8], [9], [10]. Thus, humans are capable of recognizing in a given situation what constitutes a legible trajectory. They can therefore demonstrate a legible motion to the robot so that it can make an equivalent motion for another individual. Learning from Demonstration (LfD), Programming by Demonstration (PbD) or Imitation Learning is an active area of research which enables robots to encode the demonstrated trajectories [11], [12], [13]. In LfD, it is also possible to learn a task, in this case, how to grasp an object. A non-expert in robotics can program a robot by demonstrating where to grasp an object. However, by using a motion planner, it must be programmed in order to grasp an object. At Heemskerk Innovative Technology, the robot is equipped with the Geomagic Touch haptic device with 3 DoF force feedback so that an operator can control it after deployment. Therefore, a teleoperator can provide a legible demonstration of a grasp. In this paper, the robot learns a legible grasp via teleoperated Learning from Demonstration. The approach has been tested both in simulation and on a TIAGo robot.

II. Related Work

A. Origination

Human perception research originated the concept that robot motions should convey intent. Humans rely heavily upon their ability to perceive what others are doing and to infer from gestures and expressions what they may be intended to do. According to human perception research, the brain can detect and extract intent from motions [8]. Throughout childhood, infants acquire the ability to discern intent from actions of others [14]. In addition, experiments have demonstrated that they can also extract this information from humanoid robots [15]. The infant learns to determine the intent and then use this information to predict what the human will do in the future. A collaborative task will benefit from this knowledge, as without verbal communication one can anticipate the action of the other.

B. Increase Sociability

By taking into account social constraints, human abilities, and preferences, Sisbot and Alami made a Human-Aware Manipulation Planner (HAMP) [16], [17], [18], [19], [20]. By reducing velocity and acceleration when moving near the human and respecting proxemic zones (introduced by Hall [21]), this planner is able to generate more social motions. The drawback is that, even if the above constraints are satisfied, an unclear motion without an indication of intention can cause humans discomfort.

C. Increase Intent-expressivness by Optimization

Several other techniques are used to produce an intent-expressive grasp. Dragan et al. have provided an interesting definition of legibility [2], [22]. Through the observation of a snippet of the trajectory, one is able to anticipate the goal. It is assumed that the observer (human) compares the probability of the possible goals and selects the most likely based on a portion of the motion [23]. The faster the observer correctly predicts, the more legible the motion is. At each time step, the probability of each goal is determined and placed in a cost function. Function Gradient Optimization is used to calculate a trajectory that minimizes the cost to reach the actual goal and maximizes the cost to reach other possible goals. The robot will exaggerate the motion in order to make other possible goals less likely than the correct goal. However, the motion should not be overly exaggerated, otherwise, it will be less legible. [23]. Additional advances are related to the viewpoint of the observer [24] and to expressing the incapabilities of the robot [25].
As an alternative to Functional Gradient Optimization, Bodden et al. propose a nonlinear optimization method [26], [27]. A legible motion is generated by optimizing the distance between the goal and the user's prediction at a given state, the distance between the end-effector position and the goal, and the energy. A certain heuristic is assumed in order to model the user's prediction. For example, a point position could be calculated by taking into account the current position of the

end-effector and possible goals. As a result, this Nonlinear Optimization approach generalizes better to a variety of robots and scenarios than Functional Gradient Optimization.
Lastly, some methods use Reinforcement Learning techniques [28], [29], [30], [31]. Since Reinforcement Learning involves trial-and-error, it does not comply with the concept of a robot in a highly dynamic and vulnerable environment and is therefore considered out of scope of this study.
In the described methods, objects are grasped without regard to their shape. In other words, only the motion is legible. There are, however, many objects which cannot be grasped from all angles. Thus, when the robot needs to grasp something at a specific point, the motion will be altered. As a result of considering the shape of the object, the approach is more generalizable to various situations. It is still legible when the object needs to be grasped in a specific manner. Additionally, some of these methods are not applicable to other types of robots [32], [26]. For generating a legible grasp, we recommend using Learning from Demonstration (LfD) rather than motion planners or optimization techniques. LfD has the advantage of enabling non-robotic experts to demonstrate preferred trajectories and tasks to robots. Additionally, humans are aware of what constitutes a legible motion. Furthermore, LfD is capable of generalizing to various situations.

III. Methodology

Learning from Demonstration can be divided in three main parts: the method of demonstration, the mathematical representation and learning of the low-level robot movements, and a high-level planner. For the demonstration method there are two main distinctions in Learning from Demonstration. Namely, kinesthetic teaching (the robot is physically guided by the operator) and teleoperated teaching [11]. Teleoperation provides the advantage of the operator not having to be physically present at the robot. Teleoperated demonstrations will be used in our approach since the operator will not necessarily be present in the same room as the robot.
The term "Movement Primitives" (MPs) is generally used to refer to a mathematical model for representing and learning low-level robot movements. The trajectories that are demonstrated can be encoded by Dynamic Movement Primitives (DMPs) as a nonlinear oscillator [33]. Another approach is to use Probabilistic Movement Primitives (ProMPs), which model trajectory distributions of stochastic demonstrated movements [34], [35]. The first DMP representation presented by IJspeert et al. [33] has several issues. First, when the difference between the goal and initial positions is small, the system may exhibit unexpected (and undesirable) behavior. Additionally, the motion is mirrored when the goal position minus the initial position changes sign from the learned trajectory to a new one [36]. To address these issues, a slightly different DMP representation is presented [37], [38]. DMPs have the advantage of being deterministic and are able to learn a trajectory with a single demonstration. Besides, encoding a trajectory, DMPs are capable of adapting that trajectory by updating an interactive term [39], [40]. Another extension is to plan both in joint space and Cartesian space [41]. ProMPs on the hand, generalizes better to new situations than DMPs. However, ProMPs are probabilistic and needs multiple demonstrations. Therefore, DMPs are preferred and will be used to encode trajectories.
Most often, the demonstrated data consists of kinematic coordinates, such as joint angles or Cartesian coordinates of links. However, the task (grasping the object) must also be learned. With Probabilistic Movement Primitives (ProMPs), there are probabilistic operations such as conditioning to enhance generalization to novel situations [42], [35]. With this approach, they used an external state variable, which could represent the grasped object pose. When this variable is included, the object can be placed elsewhere and still be grasped at the same position. In this paper a pre-grasp pose will be calculated from the demonstrated data and saved in a database. This is accomplished by taking the last demonstrated end-effector position and the object pose into account.
A high-level planner can be used to connect the learned motions or to let the robot choose which motion to use. It is necessary that in order to achieve a legible motion, the robot can choose a motion based on the situation. For this purpose, a Finite State Machine (FSM) or a Behaviour Tree (BT) can be used to select a motion primitive [43], [44], [45]. Since our scope revolves around the robot motion, these concepts are not considered in this study. In order to demonstrate how this works, a simple function is used to select between two different learned motion primitives. The function selects a primitive based on the environment. This primitive will be used for generalization.
In summary, a teleoperator demonstrates a legible and human-like grasp. The demonstrated data consists of the Cartesian poses in time of the end-effector and the grasped object pose. The motion is learned and encoded by DMPs as in [37] and [38]. Data for a pre-grasp pose and the learned motion is saved in a database. The result of this is that the end-effect can also be generalized to new start and end poses. A preferred grasp pose is calculated based on the demonstration to generalize the grasp to novel situations. MoveIt is used to let the robot plan and execute the motions to the new pre-grasp pose. After that, MoveIt is used to grasp the object. Figure 4 presents a general overview of the method used in this research.

A. Demonstration

The Geomagic Touch™ [2] is used as a haptic device to provide demonstrations. The operator controls with this device the 7-DOF manipulator of the care robot. The end-effector's position and orientation (6-DOF) are controlled. The inverse kinematics of the robot are taken care of by the Whole Body Controller (WBC) of the robot. The WBC is an implementation of the Stack of Tasks [46]. There are three main tasks for the WBC[3]: avoid self-collision, avoid approaching any joint limits, and following the end-effector as demonstrated with the haptic device. The WBC optimizes the tasks and establishes the joint positions (7-DOF). As a result, there is a null-space that is not controlled by the operator. The demonstrated data for the model are the Cartesian end-effector positions and

[2] https://www.3dsystems.com/haptics-devices/touch
[3] https://github.com/pal-robotics/pal_wbc_utils

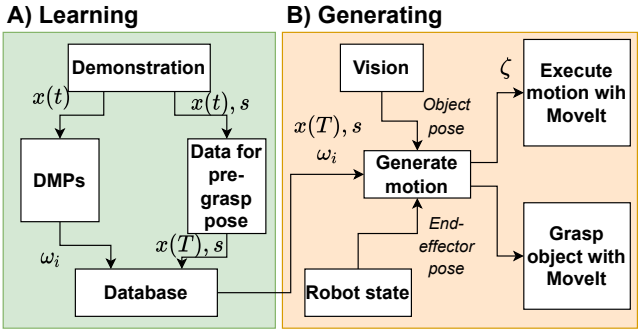


Fig. 4: Overview of the framework that is used to learn and generate a legible grasp. On the left side the learning is visualized which contains of learning a motion with DMPs and receiving data for the pre-grasp pose. On the right side is visualized how the robot can grasp the object on its own. The vision module will give the object pose and the robot the current end-effector pose. A new motion can be planned to grasp an object and MoveIt will be used to execute the motion.

orientations x in time, and the object position and orientation s at $t = 0$. Thus, $x = \begin{bmatrix} x & y & z & qx & qy & qz & qw \end{bmatrix}^T_{ee}$ and $s = \begin{bmatrix} x & y & z & qx & qy & qz & qw \end{bmatrix}^T_{object}$ at $t = 0$, with q the orientation in quaternions. After the demonstration, the Cartesian poses are manually filtered. Details are provided in Appendix A.

B. DMPs

It is possible to model both discrete and periodic movements using Dynamic Movement Primitives [33], [36]. In order to model a legible and human-like grasp, a discrete DMP model is employed. It is comprised of a system of second-order Ordinary Differential Equations (one for each dimension) of a mass-spring-damper type with a forcing term. The forcing term should be modeled by DMPs in such a way that it is able to generalize the trajectory to a new starting position and goal position for the end-effector while maintaining the shape of the learned trajectory. The following differential equation can be integrated to produce discrete movements in one dimension:

$$\begin{cases} \tau \dot{v} = K(g - x) - Dv - K(g - x_0) + Kf(s) \\ \tau \dot{x} = v \end{cases} \tag{1}$$

With $x, v \in \mathbb{R}$ the position and velocity of the end-effector. And x_0, $g \in \mathbb{R}$ the start and goal position, respectively. $\tau \in \mathbb{R}^+$ is a temporal scaling factor. The constants K and $D \in \mathbb{R}^+$ are a spring and damper constant, respectively. D is chosen to be critically damped which means that $D = 2\sqrt{K}$. $s \in (0, 1]$ is a phase variable, that is a re-parametrization of time $t \in [0, T]$. The phase is calculated by:

$$\tau \dot{s} = -\alpha s \tag{2}$$

With $\alpha \in \mathbb{R}^+$ a constant that is the exponential decay of the canonical system. The canonical system is the second equation in Equation (1). The forcing term f is a nonlinear function in order to generate complex movements and it is defined by basis functions as:

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)} \tag{3}$$

With $\psi_i(s)$ as Gaussian Basis Functions (GBFs) and is calculated by:

$$\psi_i(s) = exp(-h_i(s - c_i)^2) \tag{4}$$

with centers c_i and widths h_i. The Gaussian centers are defined by:

$$c_i = exp(-\alpha_i \frac{T}{N}), \quad i = 0, 1, ..., N \tag{5}$$

This definition enables them to be equally spaced in the time interval. The widths h_i are calculated by:

$$h_i = \frac{\tilde{h}}{(c_{i+1} - c_i)^2}, \quad i = 0, 1, ..., N - 1, \quad h_N = h_{N-1} \tag{6}$$

The overlap between the basis function can be controlled with $\tilde{h}$, which is usually $\tilde{h}$ set to one [47], [48].
Learning with DMP boils down to computing the weighs $w_i \in \mathbb{R}$ that best approximate the desired forcing term. Thus, $x(t)$ is recorded as described in the previous section, and the derivatives are computed for each time step t. The derivatives are calculated by assuming constant acceleration over a time period. The backward difference method is used to compute the derivatives. After that, $s(t)$ is computed for an appropriate adjusted temporal scaling τ, which is set to one, by:

$$s(t) = exp(-\alpha t) \tag{7}$$

The next step is to use equation (1) to calculate $f_{target}(s)$. With x_0 is $x(0)$ and g is $x(T)$, with T the last time-step. The forcing term is calculated as follows:

$$f(s(t)) = \frac{1}{g - x_0}(\dot{v}(t) - K(g - x(t)) + Dv) \tag{8}$$

Lastly, the weight vector $\vec{w_i}$ from equation (3) has to be found. It should minimize the error criterion $J = \sum_s (f_{target}(s) - f(s))^2$ [49]. This is a linear regression problem solved by weighted linear regression.
Due to the robot's end-effector's six-dimensional movement, the forcing term f has to be determined for each dimension. The system is decoupled in six dimensions in order to achieve this. This requires the creation of six decoupled copies of the system described in (1). The result is a vector formulation that is proposed in [39], [38], [50]:

$$\begin{cases} \tau \dot{\mathbf{v}} = \mathbf{K}(\mathbf{g} - \mathbf{x}) - \mathbf{D}\mathbf{v} - \mathbf{K}(\mathbf{g} - \mathbf{x}_0)s + \mathbf{K}\mathbf{f}(s) \\ \tau \dot{\mathbf{x}} = \mathbf{v} \end{cases} \tag{9}$$

With $\mathbf{x}$, $\mathbf{v}$, $\mathbf{g}$, $\mathbf{x}_0$, $\mathbf{f}(s) \in \mathbb{R}^d$ and $\mathbf{K}$, $\mathbf{D} \in \mathbb{R}^{d \times d}$ as diagonal matrices. The evolution of s is still described by the canonical system as in (3), and the forcing term $\mathbf{f}$ is still written in terms of the basis functions as in (2).

C. Generalization

The purpose of this section is to explain how generalization is accomplished to various situations. There is a new situation when an object is oriented or positioned differently. In addition, the start pose of the end-effector can be adjusted. In order to accomplish a grasp in a new situation, the robot should still grasp the object from the side as demonstrated, and the trajectory should have the same shape. In order to know where to plan, a new pre-grasp pose is first calculated.

Hereafter, a new trajectory to this pre-grasp pose is planned and executed. Finally, the object is grasped.

1) Pre-grasp Pose: The robot receives the goal object pose s_{new} from the vision module in a novel situation. Details about the vision module are described in Appendix D. It is assumed that in new situations the object can be translated in three dimensions and rotated around its yaw axis. The robot loads from the database the object pose from demonstration s_{demo} and the last pose of the demonstration $x(T)$. After that, $x(T)$ needs the same transformation as the transformation between s_{new} and s_{demo} to get the pose x_{new}. Figure 5 shows an illustration of the terms used and of the calculation steps. First, $x(T)$ is translated the same amount as the new object is translated with respect to the object used in the demonstration. So the translation is $\mathbf{v} = s_{new} - s_{demo}$, and therefore the translation of the pre-grasp pose is $x_{translation} = x(T) + \mathbf{v}$. Next, the rotation of the object must be taken into account. For that, the yaw angle ψ between the demonstrated object and the new object is determined. The new pre-grasp pose is $x_{new} = \mathbf{R}x_{translation}$. With $\mathbf{R}$ the rotation matrix with angle ψ.

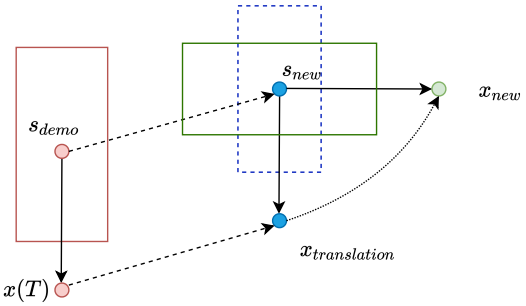


Fig. 5: Calculating a new pre-grasp pose. With s_{demo} the object pose as demonstrated and s_{new} the object pose in a new situation. First, the pre-grasp pose $x(T)$ is translated with the same translation as between the object poses to $x_{translation}$. The new pre-grasp pose x_{new} is calculated by taking into account the rotation of the object.

2) Trajectory: The new pre-grasp pose makes it possible for the DMPs to generate a new motion with the same shape as demonstrated. The weight function from the database is loaded to calculate the forcing terms $\mathbf{f}$, the new goal position g represents the pre-grasp pose x_{new}, and x_0 represents the current end-effector pose of the robot. The parameters $\mathbf{K}$ and $\mathbf{D}$ are the same as those used in the learning procedure. The temporal scaling factor τ is the desired length of the total DMP execution in seconds. Equation (7) and τ are used to determine $s(t)$. By completing Equation (9), a new acceleration $\dot{\mathbf{v}}$ is derived. The new positions and velocities are calculated using Euler Integration: $\mathbf{v} = \dot{\mathbf{v}}dt$ and $\mathbf{x} = \dot{\mathbf{x}}dt$. A new trajectory with a different goal pose and the same shape as the previous one is depicted in Figure 6. The blue dot represents the start pose of the end-effector, the orange dot represents the pre-grasp pose used by demonstration, the green trajectory represents the learned motion based on the demonstration, and the green dot and blue trajectory represent the new pre-grasp pose and generalized trajectory, respectively.

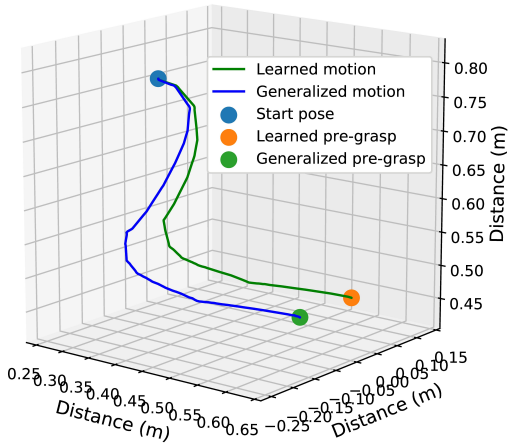


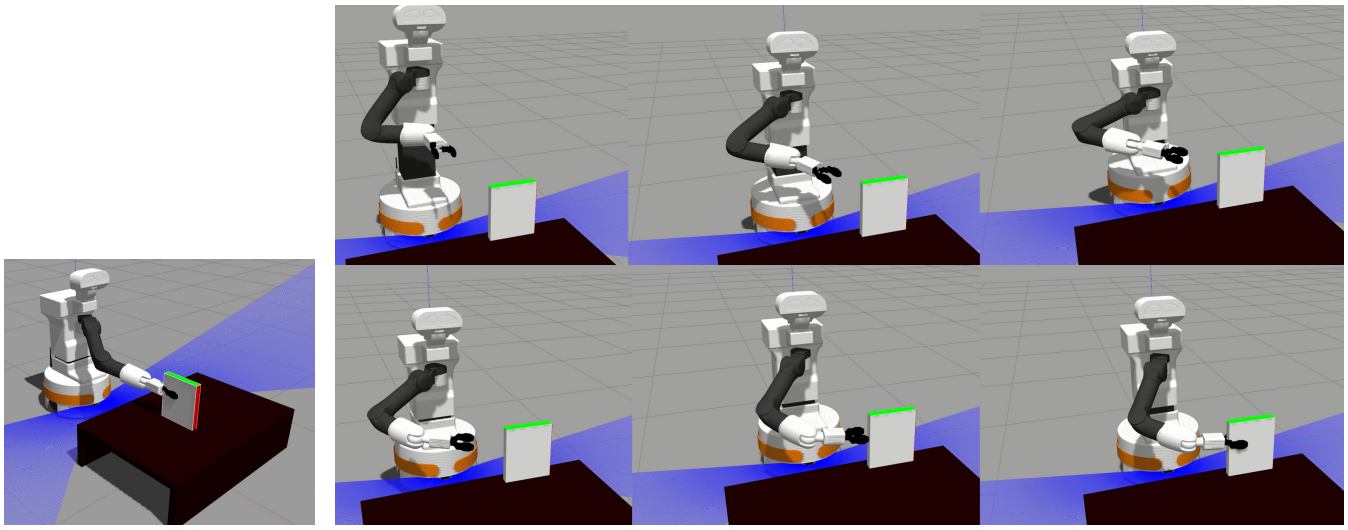
Fig. 6: Generalization to a different object pose. The position of the end-effector is the same for both cases (blue dot). An operator has demonstrated the blue path, and the initial pre-grasp pose was located at the green dot. Subsequently, the object was positioned differently. The method was able to calculate a new pre-grasp pose (orange dot) as well as a new path (green line) that has the same shape as demonstrated.

IV. Design Evaluation Experiment

To demonstrate the effectiveness of the proposed teleoperated learning method, the method is initially implemented in a 3D simulation environment. In addition, how well the method generalizes to various new object poses and varying initial poses of the end-effector is tested in simulation. To validate the designed method, it is compared to an off-the-shelf motion planner on success-rate, velocity, acceleration and smoothness.

A. Simulation

In order to demonstrate that the method is valid in simulation, an object is modelled and placed on a table. To test the method a cereal box is used as an object and it is placed on a table. The TIAGo robot is selected as robot to grasp this object. After demonstrating a grasp, the robot learns the motion and calculates a pre-grasp pose. The same cereal box is then placed on the table. In order to grasp the box the robot has to know the motion primitive, the pre-grasp pose, the object pose, and the initial end-effector pose. The robot loads the motion primitive and the pre-grasp pose from the database. The cereal box's pose is determined by the implemented vision module which uses ArUco markers (Appendix D). In order for the TIAGo robot to grasp a cereal box on its own, MoveIt is used to execute the learned grasp. See Appendix B for more details about MoveIt. MoveIt returns also the initial end-effector pose. In such situations, the robot should be able to calculate a new pre-grasp pose, move towards it with the same shape of motion as demonstrated and grasp the cereal box. Appendix C contains the results of the simulation involving the grasping of a cereal box. Following a single teleoperated demonstration and the calculation of a new pre-grasp pose, the robot was able to generalize the motion and grasp the cereal box in various poses. Figure 7(a) illustrates how the cereal box was grasped when the box is placed in the same pose as demonstrated. Figure 7(b)



(a) Grasping box without generalization

(b) Grasping box with generalization

Fig. 7: **Left**: Grasping a cereal box in simulation, with the cereal box placed at the same pose as in the demonstration. The robot was able to learn the motion and grasp the cereal box again. **Right**: Grasping motion of the cereal box that is placed in a different pose than in the demonstration. The robot is able to calculate a new pre-grasp pose and generalize the motion towards that pre-grasp pose. Hereafter, the robot could grasp the box successfully.

illustrates how the robot grasps the cereal box with the same motion, but with a different orientation and position of the cereal box.

B. TIAGo Robot

Several expectations will be tested and evaluated with the real TIAGo robot. The main difference with the simulation is that there could be more noise (e.g. noise from the camera). Therefore, the first test will determine if the robot is capable of automatically grasping objects after having learned to grasp an object. Next the velocity, acceleration and smoothness will be evaluated because it is related to legibility [1]. According to [16], [17], [18], [19], [20] a more social motion is already better for the human perception. A more social motion is one that has lower velocities, accelerations and a better smoothness. Despite MoveIt controlling the velocity and acceleration of both the LfD method and the RRTConnect motion planner, the velocity and acceleration could still differ.

In general, grasping is achieved through motion planners or machine learning. However, machine learning requires a large amount of data, whereas motion planners are available off-the-shelf in MoveIt. Thus, our LfD framework will be compared to an off-the-shelf motion planner from the MoveIt library. In [51], they compared all the motion planners available in MoveIt with the UR5 robot. For different grasp executions, BKPiece and RRTConnect provided the most consistent results with this robot, RRTConnect performed better in terms of path length (shorter path). Therefore, the method will be compared with the RRTConnect motion planner.

1) Test Setup: During the tests, an object will be placed on a table. Again a cereal box will be used to demonstrate the effectiveness. The experimental setup is illustrated in Figure 8. The robot (orange oval) will be positioned near a table (gray rectangle). The same cereal box will be placed at random on the table (green rectangles). A single teleoperated demonstration will allow the robot to learn how to grasp a cereal box (e.g. yellow rectangle). After the cereal box has been grasped with the teleoperated learning method, it will be placed in the same poses and grasped with the RRTConnect motion planner. The start pose of the end-effector will be the same for both methods.

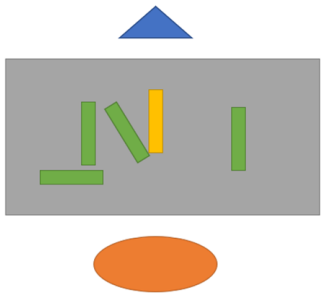


Fig. 8: Setup for evaluating the designed method. The orange oval is the robot base, the grey rectangle the table, the yellow rectangle the object used for the demonstration, and the green rectangles the same object in different poses. The blue triangle is the place where the participant will observer the robot.

2) Procedure: First, the operator will demonstrate how the robot should grasp the object. The robot will then learn the motion and calculate the pre-grasp pose. Afterwards, the cereal box will be placed randomly in nine different poses on a table within the robot's workspace, and the robot should be able to pick up the box. The box will be placed at the same position as in the demonstrated data, to the right and to the left, rotated by approximately $\pm$ 90 degrees and

$\pm$ 45 degrees, and higher and closer to the robot than used by the demonstration. The robot will grasp the box with the learned motion and with the RRTConnect motion planner. Data received from the robot is the pose in time of the different frames of the robot arm. This data comes from the implementation of the tf package [52].

3) Expectations: During the first evaluation, it will be investigated whether teleoperated LfD is suitable for grasping objects with a real robot. We expect that the robot can learn to grasp objects by using teleoperated learning from a single demonstration.
Second, we investigate whether the executed motion is already more socially constrained. It is expected that the maximum and mean velocity and acceleration of the end-effector are lower with the LfD method. Lower speeds and accelerations make humans feel more comfortable around the robot [20]. Lastly, we expect that the learned motion is smoother than the motion from the RRTConnect motion planner. A smoother motion is perceived as more comfortable for the human [1].

4) Metrics: The success rate of the grasp will constitute the primary metric for the first expectation. Thus, it is determined whether the robot is capable of grasping the object. Furthermore, it is necessary to determine whether it collides with the environment (static objects) or with the object to be grabbed.
Additionally, the maximum and mean velocity and acceleration can be determined by computing the first and second derivatives of the positional data, respectively. This is accomplished using the numerical central difference method. Smoothness is often compared by calculating the jerk which is the third derivative of the position with respect to time. A lower jerk is in most cases a smoother motion. However, noise and path-length can influence the jerk. Therefore, we want to combine jerk and the zero-crossings of the velocity to compare smoothness. Less zero-crossings indicate that the motion is more straightforward for the perception of humans. It has not to be zero, because when avoiding objects there are more zero-crossings. However, in the same situation the zero-crossings should be as low as possible. It is calculated by first finding the points where the velocity is zero. After that, the zeros where the velocity are not changing sign are deleted. The result is that there are only points where the velocity is going through zero. For the jerk a value is needed to compare the methods with. Therefore, the integrated square jerk of the end-effector and of the elbow motion will be calculated. First, the jerk is squared to ensure a positive outcome. The result is that by integrating the jerk the negative jerk will not cancel out the positive jerk. The equation for the integrated square jerk is:

$$J = \int_{t_1}^{t_2} \dddot{x}(t)^2 \, dt \tag{10}$$

with $\dddot{x}(t)$ the third derivative of position in time. The motion is in general smoother when the integrated square jerk is lower.
The RRTConnect motion planner can get different solutions for the same scenario, because it is not an optimal planner. To account for this, the box will be grasped two times in each object pose with the MoveIt motion planner. In simulation we have seen that with the LfD method the motion is always the same, therefore it is not needed to account for different motions.

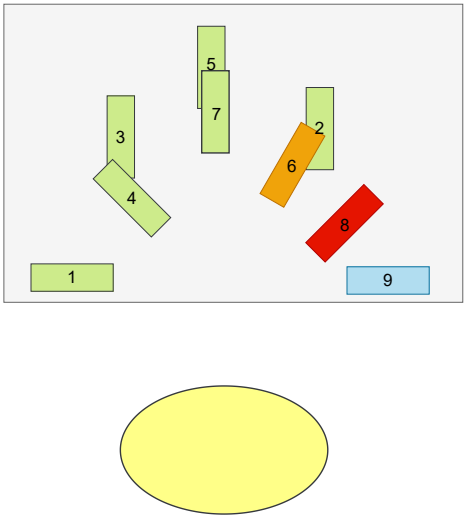


Fig. 9: The nine different cereal box poses on the table that are tested. Box 7 has a higher z-coordinate than the rest of the boxes. The pose of box 5 is the same as when the operator demonstrated the grasp. The green cereal boxes are grasped with both the LfD method as with the RRTConnect motion planner. The orange box could only be grasped with the LfD method. The motion planner collided with the cereal box. The blue box is only successfully grasped with the RRTConnect motion planner and the red box could not be grasped with both methods. The yellow oval is the base of the robot.

5) Results Grasping: In Figure 9 the results of grasping the cereal box in nine various poses are illustrated. The green boxes are the cereal boxes that could be grasped with both methods. The orange box could only be grasped with the learning method, while the blue box could only be grasped with the RRTConnect motion planner. Both methods could not successfully grasp the red box. From this we can see that with both the teleoperated LfD method and the RRTConnect motion planner the robot was able to successfully grasp the box in seven different poses. The failures are further analyzed to see what the reason is that the robot could not grasp the box. With the learning the robot was not able to successfully grasp the cereal box in pose 8 and 9 (red and blue rectangles in Figure 9). For pose 8 we visually analysed that the robot had problems with the configuration of the arm when trying to follow the path. The start configuration of the arm is shown in Figure 10. By following the way-points the end-effector needs to move in such a way that the elbow has to flip to the other side (left handed). However, the robot is constrained not to do so. Therefore, the robot gets stuck due to the joint limits and cannot execute this motion. It is harder for the robot to move the yaw angle of the end-effector counter clockwise. Therefore, it is impossible for the robot to execute a motion when the DMPs generalizes to a motion that plans to let the end-effector move like this. A solution for this is to start in a left handed configuration. In that way the robot is able to

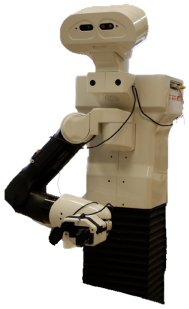


Fig. 10: Start configuration of the robot arm. The end-effector can easily rotate the yaw angle counter clockwise. It is hard to rotate it clockwise.

execute the motion and grasp the cereal box with the learned motion.

Pose 9 is where the yaw angle of the box is 90 degrees with respect to the one in the demonstration (blue rectangle in Figure 9). The problem with grasping this box is the same as for Pose 8. However, when starting in a different start configuration there is still a problem that the robot will do a motion that is exaggerated to one side. Due to this the end-effector will collide with the cereal box and therefore the motion cannot be executed. This demonstrates that with the LfD method it is probably safe to execute motions. An image of the visualisation of the virtual environment is shown in Figure 11. By following the learned motion and getting a good orientation to grasp the box the robot will collide with the object. A solution for this case could be to use a different motion. A better solution could be to replace the base of the robot so it will be easier for the robot to grasp the box. Another solution could be to grasp from the topside of the cereal box.

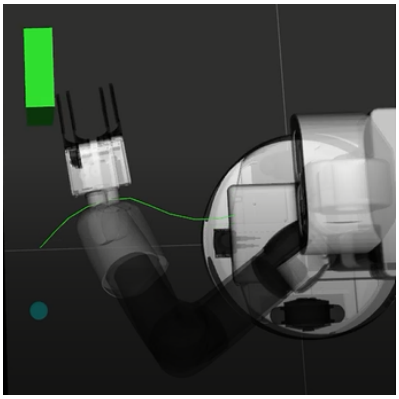


Fig. 11: Top view of the robot in the virtual environment. It shows that the robot will collide with the box (green rectangle) when following the motion (green path). Another motion could probably solve this issue.

The RRTConnect motion planner could not grasp the box when it was placed in pose 6. The end-effector collided with the cereal box before grasping. In Figure 12 is shown how the robot collides with the cereal box in the virtual environment. Due defining the pre-grasp pose too close to the cereal box, MoveIt makes a plan that collides with the box. Apparently, MoveIt does not check collision with the

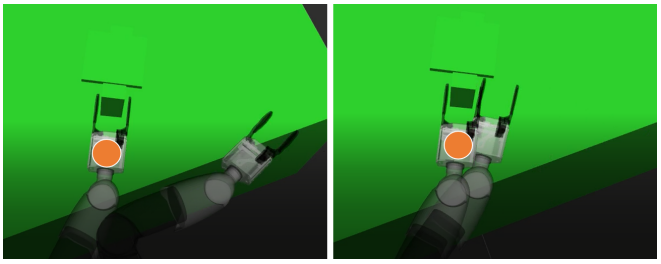
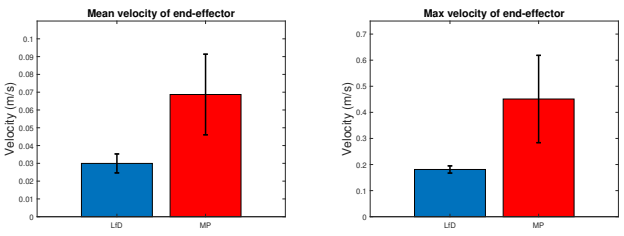


Fig. 12: Top view of the table and cereal box with marker in a virtual environment. The left side shows how the end-effector is approaching the box. The orange circle represents the pre-grasp pose. In the right figure the end-effector collides with the box.

grasped object when moving to the pre-grasp pose. The pre-grasp pose should be defined in such a way that the gripper cannot touch the box.
For pose 8 (red rectangle) MoveIt could not find a plan in time. Even after giving multiple times the command to find a solution the robot was not able to find a plan.
Concluding, the robot was able to grasp objects by learning from a single teleoperated demonstration. The robot was able to autonomously grasp the cereal box in 7 out of 9 poses which is the same with the RRTConnect motion planner. Objects that are clockwise rotated (yaw angle) are harder or impossible for the robot to grasp with a exaggerated motion. For that a different start configuration, different base position or a new learned motion might solve this. A high-level planner might solve some of those issues.

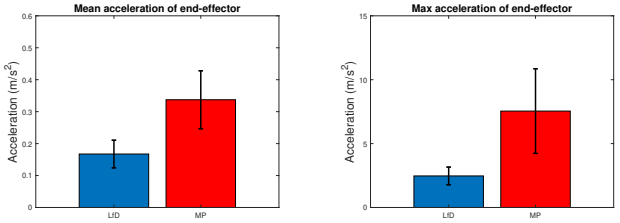
6) Results Sociability: For calculating the velocities, accelerations and smoothness of the executed motion the same poses of the boxes and end-effector are used as described in previous subsection. The boxes that could be grasped with one method and not with the other are not taken into account. We will go over the results one by one in this subsection. First, the results of the velocity will be presented and discussed. Next, the accelerations of the end-effector. After that, the smoothness will be elaborated. Due to the noise in the data an zero-phase digital low-pass filter is used to reduce the noise and still have zero phase distortion. The reason is that if there is noise in the data from the robot it will increase when calculating the derivatives.
In Figure 13(a) and 13(b) the max and mean velocity of both methods is compared, respectively. The histogram shows



(a) Mean velocity of end-effector (b) Max velocity of end-effector

Fig. 13: The mean and max velocity for both the LfD method as for the RRTConnect motion planner. The mean of both methods with the standard deviation is plotted. It shows that in both cases the velocity is lower with the LfD method.

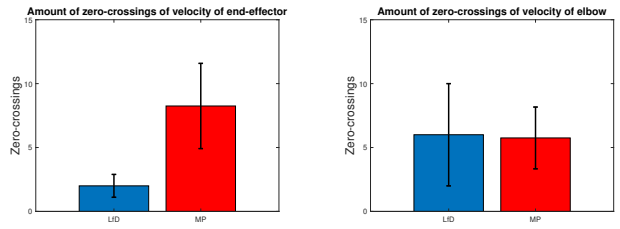
the mean of both methods and the standard deviation. In both cases the LfD method has a lower mean and a smaller deviation. An explanation for this could be that by following the waypoints MoveIt has to control the joints more precisely. This is easier to accomplish when the velocity is lower. It seems like the velocity is already constrained with the LfD method. Adding constraints to the velocity is probably not necessary, however to ensure that it is always lower in close proximity it could be added in the implementation.
The mean and max accelerations are plotted in Figure 14(a) and 14(b). For the accelerations it is the same as with the velocity. By using a different method in MoveIt the end-effector is possibly more constrained. With the learning the robot moves from point to point. It could be that the robot has not the time to fully accelerate. Which might result in lower accelerations. MoveIt has an option to manually constrain the accelerations. It might help to add this when the end-effector moves closer to humans.



(a) Mean acceleration of end-effector (b) Max acceleration of end-effector

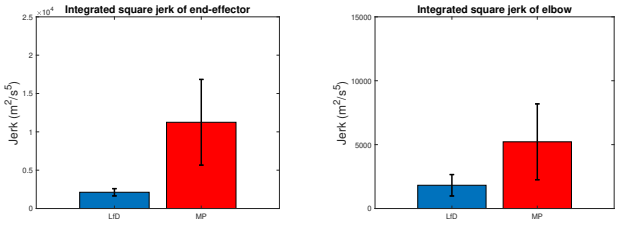
Fig. 14: The mean and max acceleration for both the LfD method as for the RRTConnect motion planner. The mean of both methods with the standard deviation is plotted. It shows that in both cases the acceleration is lower with the LfD method.

For the smoothness both the zero-crossings of the end-effector and elbow and are plotted in Figure 15(a) and Figure 15(b), respectively. Furthermore, Figures 16(a) and 16(b) shows the integrated square jerk of the end-effector and elbow, respectively. The end-effector motion has less zero-crossings and a lower jerk with the LfD method. However, the elbow motion has a higher mean and standard deviation in the zero-crossings than the RRTConnect motion planner. The end-effector motion is only learned and the operator has probably shown a motion that is more straightforward for the end-effector. It could be that the null-space of the robot



(a) Zero-crossings of velocity end-effector (b) Zero-crossings of velocity elbow

Fig. 15: Zero-crossings of the velocity of both the end-effector and elbow. A zero-crossing means that the direction of the frame changes. Less zero-crossings could mean that the motion is smoother and that it is more clear for a human what the robot is doing.



(a) Integrated square jerk of the end-effector (b) Integrated square jerk of the elbow

Fig. 16: Integrated square jerk of both end-effector and elbow frame. The integrated square jerk is a measure for smoothness of a motion. The lower the jerk, the smoother a motion will be in general. However, other aspects like path-length could influence this measure.

should also be considered in generating a legible grasp. From the combination of both the zero-crossings of the velocity and the integrated square jerk we can see that the end-effector motion is probably smoother. However, the null-space motion of the robot is not necessarily smoother. It could be that the robot needs to move the elbow more to follow the defined waypoints. It might be interesting to investigate how the whole arm of the robot should behave in an environment with a human close to a robot.
We tried to give some measures that are used for legibility or aspects that are related to legibility in this subsection. From all this data we can conclude that the robot was more constrained with the teleoperated LfD method in these grasps. The robot moves more social because the velocity and accelerations are lower. Besides, the end-effector motion is smoother than with the RRTConnect motion planner in those cases. Altogether it might suggest that the motion is perceived more legible for a human. To validate more the legibility of the designed teleoperated LfD module an experiment with human participants is designed in the next section.

V. Human Factor Experiment: Legibility of the Executed Motion

In the previous section the executed motion was analyzed from a more technical perspective. The goal of this experiment is whether the teleoperated LfD method is more legible than the RRTConnect motion planner in terms of intent-expressiveness. For this an experiment with humans observing the robot's motion will be conducted. The learned motion will again be compared to the RRTConnect motion planner from MoveIt.

1) Test Setup: The experimental setup is illustrated in Figure 17. In this setup two identical cereal boxes are placed on a table. The robot is positioned near the table and a video camera is placed at the other side. The robot will start always in the same start position.

2) Procedure: Initially, the robot will learn two different motions, one that is more exaggerated to the right and one that is more exaggerated to the left. Based on the pose of the other object, the robot will select one of the motion primitives to indicate the intent. Whenever the goal object

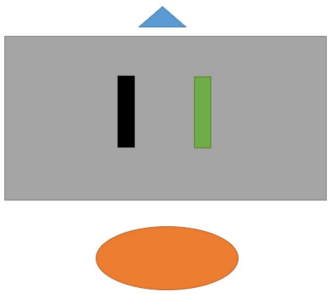


Fig. 17: Setup for the experiment, the black and green rectangles are identical boxes placed randomly on the table. The blue rectangle is the position of the camera.

is placed to the left of the other object, the robot will choose a motion that exaggerates to the left. The robot will exaggerate to the right otherwise. The RRTConnect motion planner will perform the same task. The motions are recorded by a camera from the opposite direction (blue rectangle in Figure 17). The motions are recorded for seven different combination of poses. After that, participants will see a movie showing a part of the robot's motion when it grasps one of the boxes[4]. This will be either with the RRTConnect MoveIt motion planner, or with the LfD method. The participant, however, will not know which object the robot will grasp. Only a few seconds of the robot's motion are shown in the movie, and participants will be asked to make a prediction. The time when the video stops can be different for a different object pose. However, the time is the same for the RRTConnect motion planner as for the LfD method when the objects are in the same poses. Figure 18 illustrates how one video can look like. The left side shows the start pose of the robot and the poses of the boxes in this situation. The right image could be the frame after a few seconds of execution when the video stops.

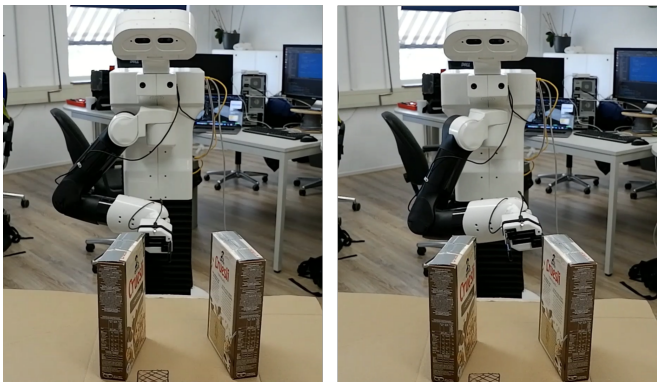


Fig. 18: The left figure shows the start of a short video. The right side shows an example of the frame when the video stops. The two boxes can be positioned both in a different pose.

3) Expectations: The teleoperator should be able to demonstrate a motion with intent. The robot should be able to learn that motion and generalize that motion to new situations. The RRTConnect motion planner from MoveIt will probably give not any signs of intent. Therefore, the expectation is that the human will guess the intention of the robot. We expect that the intention of the executed motion can be better inferred by a human with the teleoperated learning method. Since the correctness and certainty of the prediction are essential, the main expectation is divided into two more specific expectations. We expect that humans are more likely to make a correct prediction with the LfD method than with the RRTConnect motion planner. When participants are guessing they will also be less certain of their prediction. It is expected that they will guess less with the teleoperated LfD method and therefore are more certain about the prediction. Therefore, we expect that the teleoperated LfD method allows the human to make a more certain prediction.

4) Metrics: The first metric will be a correct prediction of the goal object. This will be either a correct or incorrect prediction. The second metric will be the certainty on a 5-point Likert scale. The questionnaire given to the participants can be found in Appendix F. After a pilot recording, it was observed that the executed motion with the RRTConnect motion planner was more aggressive. Therefore, the path of the end-effector as well as the path of the elbow was recorded in time. Analyzing that data might give some insight to the answers of participants.

5) Results Human Prediction: The experiment was conducted by nine participants with a technical background. They have seen 16 different short videos with the robot executing a part of the motion. In half of the videos the robot used the RRTConnect motion planner and the other half the learning method. The robot grasped a cereal box in 7 various poses with both methods. One extra situation is created by showing the participant the same video but with a longer duration, so they saw first a short part of the motion. Later on they saw the same motion but was stopped at a later time. The results of the correctness of both methods is visualized in Figure 19. From this figure it can be quickly inferred that with the LfD method the participants could make a correct prediction by seeing a part of the motion. While with the RRTConnect planner the participants are probably guessing the answer. The mean is even under the 50 percent, which might indicate that the robot gives a wrong impression. Because the data is discrete, dependent and not necessarily normally distributed a Friedman test with a post hoc analysis could be utilized to check the first expectation. With the most conservative post hoc analysis (Bonferroni) the significance is P=0.0102. The result is smaller than P<0.05 and therefore we can conclude that there is a significant difference. Concluding, that humans are more likely to make a correct prediction with the teleoperated LfD method than with the RRTConnect motion planner.
The results of the certainty per method is presented in Figure 20. The difference is again large between the two methods. The data for the certainty is discrete, categorical (ordinal),

[4]The video shown to participants can be found here: `https://youtu.be/2m4YLzLJuw0`

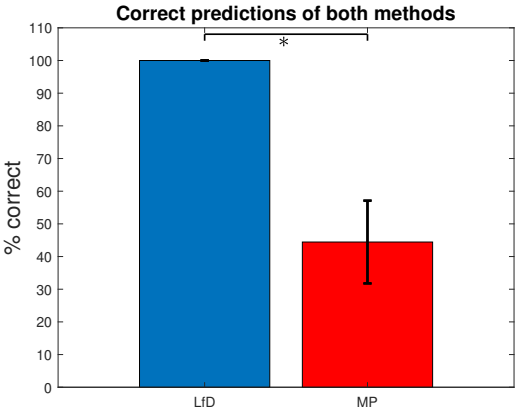


Fig. 19: Percentage of correct predictions, with MP representing the RRTConnect motion planner from MoveIt. With the height of the bars the mean of both methods and the error bars represents the mean plus and minus the standard deviation. With the significance between data sets: $\star$ P<0.05, $\star\star$ P<0.01, $\star\star\star$ P<0.001.

could be dependent because the participants have seen both methods, and is not necessarily normally distributed. Therefore, again a Friedman test is utilized with a post hoc analysis to calculate the significance. With the post hoc analysis the result is P=0.0002. The second expectation was that we expect that the teleoperated LfD method allows the human to make a more certain prediction. The calculations show that there is a significant difference between the data. Therefore, we can conclude that participants are more certain about their predictions when the robot execute a learned motion. The data shows a large variance in the certainty by the RRTConnect motion planner. It could be that it is different per question or that a more confident participant gives a higher rating.

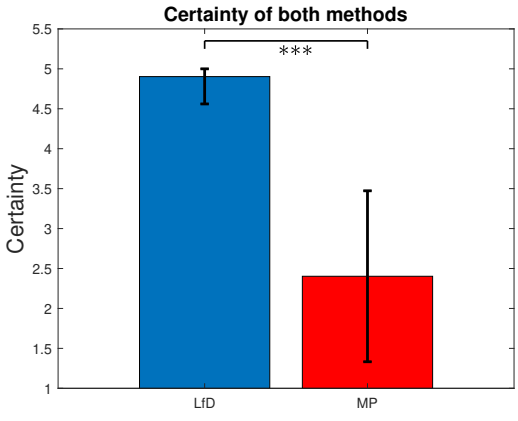


Fig. 20: Certainty on a 5 point Likert scale, with MP representing the RRTConnect motion planner from MoveIt. With the height of the bars the mean of both methods and the error bars represents the mean plus and minus the standard deviation. With the significance between data sets: $\star$ P<0.05, $\star\star$ P<0.01, $\star\star\star$ P<0.001.

The RRTConnect motion planner data per question is presented in Table I. Extra information, like path length, velocities, acceleration and integrated square jerk of the end-effector and elbow frame of all the motions can be found in Appendix E (Table E.1 and Table E.2, respectively). It is noticeable that question 13 is wrongly answered by all participants. In video 13 and 5 the same motion was showed with a different duration. The robot moved with the RRTConnect motion planner first to the right side of the robot, stopped, moved towards the boxes, passed one box and grasped the other. Video 5 was stopped just after the robot starts moving towards the boxes. Video 13 was stopped when the robot approached the other object. The robot gave the wrong impression of what is was doing.
Video 7 has the highest mean certainty and the lowest standard deviation. By analyzing the video it is observed that the video stops exactly at the point that the end-effector passed one of the boxes (not the goal object). This could be the reason that the certainty increased. Still some participants thought the robot would grasp the other box. However, they were also less certain about their prediction.
Something else worth to mention is video 14 were all participants gave a correct answer. However, they were quite uncertain about their prediction. Table E.1 from Appendix E shows that the integrated square jerk of the end-effector was the lowest from all the motions of the RRTConnect motion planner. Besides, Table E.2 from Appendix E shows that the robot moved the elbow different than the other motions from the RRTConnect motion planner. The mean velocity of the elbow, the zero-crossings of the velocity and the integrated square jerk are the lowest with this motion. It might suggest that not only the end-effector of the robot is expressing intent but also the null-space of the robot. However, further research could be done in null-space movement and legibility.
Participants were shown a large portion of the robot's movement. By doing so, we wanted to give participants an opportunity to infer the intent of the robot as it executed a motion using the RRTConnect motion planner. However, the videos could also be cut off earlier in order to test whether the participants were able to deduce the intent by observing a shorter portion of the movement. It would be interesting to investigate how fast humans infer intent from other humans and use that time for robots as well.

VI. Conclusion

In this research, we showed how teleoperated Learning from Demonstration can be utilized to achieve a more legible grasp than general motion planning with a TIAGo robot. It consists of modelling a single teleoperated demonstration using Dynamic Movement Primitives and calculating a grasping pose for the object. With a single demonstration, it was demonstrated that the robot is capable of learning and performing a grasp in various object poses. Moreover, the executed motions was smoother, and the velocity and acceleration were more constrained. Thereafter, an experiment demonstrated that with the learning method humans were able to better infer the robot's intent from an observation of a part of its motion than with a state-of-the-art motion planner. The results revealed that the operator could convey intent to the robot and the robot was able to discern it. As a first step, this research shows that teleoperated LFD can be used to learn tasks. Besides, it shows that it is more legible for humans in the robot's environment when the robot is

TABLE I: In this table the data of the motion planner is presented. The correct prediction per questions is shown with the mean certainty of the participants.

Place video	Duration (s)	Correct prediction (%)	Mean certainty	Std. deviation certainty
2	4.0	56	2.56	1.13
3	3.0	33	2.11	1.17
5	5.0	11	2.44	1.01
7	5.0	67	3.00	0.87
8	4.0	22	1.89	0.93
12	3.0	67	2.44	1.33
13	7.0	0	2.56	1.13
14	4.5	100	2.22	0.97

automated to grasp objects.

During this study, the operator was only able to control the robot's end-effector position. Possibly further research could be conducted on controlling the null-space of the robot and learning in joint and cartesian space. We have seen that the null-space motion is less smooth and still has some unpredictability in it. DMPs are used to model the end-effector path, however, the base position of the robot relative to the object should also be considered. Lastly, a high-level planner could be added to solve issues as choosing the start configuration of the end-effector or choosing different motion primitives. With those recommendations, it is expected that the robot will be able to grasp objects in a more legible manner in even more situations.

REFERENCES

[1] C. Lichtenthäler and A. Kirsch, "Legibility of Robot Behavior : A Literature Review," Apr. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01306977

[2] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 301–308, 2013.

[3] P. Basili, M. Huber, O. Kourakos, T. Lorenz, T. Brandt, S. Hirche, and S. Glasauer, "Inferring the goal of an approaching agent: A human-robot study," *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 527–532, 2012.

[4] L. Takayama, D. Dooley, and W. Ju, "Expressing thought: Improving robot readability with animation principles," *HRI 2011 - Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction*, no. January 2011, pp. 69–76, 2011.

[5] C. L. Nehaniv, K. Dautenhahn, J. Kubacki, M. Haegele, C. Parlitz, and R. Alami, "A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction," *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, vol. 2005, pp. 371–377, 2005.

[6] A. Sciutti, C. Ansuini, C. Becchio, and G. Sandini, "Investigating the ability to read others' intentions using humanoid robots," *Frontiers in Psychology*, vol. 6, 9 2015.

[7] A. D. Dragan and S. S. Srinivasa, "Familiarization to robot motion," *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 366–373, 2014.

[8] S. J. Blakemore and J. Decety, "From the perception of action to the understanding of intention," *Nature Reviews Neuroscience*, vol. 2, no. 8, pp. 561–567, 2001.

[9] L. Sartori, C. Becchio, and U. Castiello, "Cues to intention: The role of movement information," *Cognition*, vol. 119, no. 2, pp. 242–252, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.cognition.2011.01.014

[10] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: Bodies and minds moving together," *Trends in Cognitive Sciences*, vol. 10, no. 2, pp. 70–76, 2006.

[11] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Robot Learning from Demonstration: A Review of Recent Advances," vol. 3, pp. 1–33, 2020. [Online]. Available: www.annualreviews.org

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

[13] T. Osa, G. Neumann, J. Pajarinen, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *arXiv*, vol. 7, no. 1, pp. 1–179, 2018.

[14] G. Gergely, Z. Nádasdy, G. Csibra, and S. Bíró, "Taking the intentional stance at 12 months of age," *Cognition*, vol. 56, no. 2, pp. 165–193, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/001002779500661H

[15] K. Kamewari, M. Kato, T. Kanda, H. Ishiguro, and K. Hiraki, "Six-and-a-half-month-old children positively attribute goals to human action and to humanoid-robot motion," *Cognitive Development*, vol. 20, no. 2, pp. 303–320, 2005.

[16] E. A. Sisbot, L. F. Marin, R. Alami, and T. Simeon, "A mobile robot that performs human acceptable motions," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1811–1816, 2006.

[17] E. A. Sisbot, L. F. Marin, and R. Alami, "Spatial reasoning for human robot interaction," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2281–2287, 2007.

[18] E. A. Sisbot, K. F. Marin-Urias, R. Alami, and T. Siméon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[19] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila, "Toward human-aware robot task planning," *AAAI Spring Symposium - Technical Report*, vol. SS-06-07, pp. 39–46, 2006.

[20] E. A. Sisbot, L. F. Marin-Urias, X. Broquère, D. Sidobre, and R. Alami, "Synthesizing robot motions adapted to human presence: A planning and control framework for safe and socially acceptable robot motions," *International Journal of Social Robotics*, vol. 2, no. 3, pp. 329–343, 2010.

[21] E. T. Hall, *The hidden dimension*. Chicago: Doubleday Company, 1966.

[22] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[23] A. Dragan and S. Srinivasa, "Integrating human observer inferences into robot motion planning," *Autonomous Robots*, vol. 37, no. 4, pp. 351–368, 2014.

[24] A. D. Dragan, K. Muelling, J. Andrew Bagnell, and S. S. Srinivasa, "Movement primitives via optimization," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 2339–2346, 2015.

[25] M. Kwon, S. H. Huang, and A. D. Dragan, "Expressing Robot Incapability," *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 87–95, 2018.

[26] C. Bodden, D. Rakita, B. Mutlu, and M. Gleicher, "Evaluating intent-expressive robot arm motion," *25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016*, pp. 658–663, 2016.

[27] C. Boden, D. Rikita, B. Mutlu, and M. Gleicher, "A flexible optimization-based method for synthesizing intent-expressive robot arm motion," *International Journal of Robotics Research*, vol. 37, no. 11, pp. 1376–1394, 2018.

[28] B. Busch, J. Grizou, M. Lopes, and F. Stulp, "Learning Legible Motion from Human–Robot Interactions," *International Journal of Social Robotics*, vol. 211, no. 3-4, pp. 517 – 530, Mar. 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01629451

[29] F. Stulp, J. Grizou, B. Busch, and M. Lopes, "Facilitating intention prediction for humans by optimizing robot motions," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 1249–1255, 2015.

[30] K. Hoang Dinh, O. S. Oguz, M. Elsayed, and D. Wollherr, "Adaptation and Transfer of Robot Motion Policies for Close Proximity Human-

Robot Interaction," *Frontiers in Robotics and AI*, vol. 6, no. July, 2019.

[31] M. Bied and M. Chetouani, "Integrating an Observer in Interactive Reinforcement Learning to Learn Legible Trajectories," *29th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2020*, pp. 760–767, 2020.

[32] J. Zhao, B. Xie, and C. Song, "Generating human-like movements for robotic arms," *Mechanism and Machine Theory*, vol. 81, pp. 107–128, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.mechmachtheory.2014.06.015

[33] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," vol. 2, 02 2002, pp. 1398 – 1403.

[34] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

[35] ——, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[36] M. Ginesi, N. Sansonetto, and P. Fiorini, "Overcoming some drawbacks of Dynamic Movement Primitives," *Robotics and Autonomous Systems*, vol. 144, 2021.

[37] P. P. Dae-Hyung Park, Heiko Hoffmann and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on, IEEE*, pp. 469–474, 2008.

[38] H. Hoffmann, P. Pastor, D. H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2587–2592, 2009.

[39] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, 2009.

[40] M. Chi, Y. Yao, Y. Liu, and M. Zhong, "Learning, generalization, and obstacle avoidance with dynamic movement primitives and dynamic potential fields," *Applied Sciences (Switzerland)*, vol. 9, no. 8, 2019.

[41] C. Lauretti, F. Cordella, and L. Zollo, "A Hybrid Joint/Cartesian DMP-Based Approach for Obstacle Avoidance of Anthropomorphic Assistive Robots," *International Journal of Social Robotics*, vol. 11, no. 5, pp. 783–796, 2019. [Online]. Available: https://doi.org/10.1007/s12369-019-00597-w

[42] M. Ewerton, G. Maeda, G. Kollegger, J. Wiemeyer, and J. Peters, "Incremental imitation learning of context-dependent motor skills," *IEEE-RAS International Conference on Humanoid Robots*, pp. 351–358, 2016.

[43] K. French, S. Wu, T. Pan, Z. Zhou, and O. C. Jenkins, "Learning behavior trees from demonstration," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 7791–7797, 2019.

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

[45] M. Colledanchise and P. Ogren, "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 372–389, 2017.

[46] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," *2009 International Conference on Advanced Robotics, ICAR 2009*, no. 8, 2009.

[47] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in Cartesian space dynamic movement primitives," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 3, pp. 2997–3004, 2014.

[48] F. J. Abu-Dakka and V. Kyrki, "Geometry-aware Dynamic Movement Primitives," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 0, pp. 4421–4426, 2020.

[49] R. Mao, "Robots Learning Manipulation Tasks," p. 110, 2016.

[50] D. H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," *2008 8th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2008*, no. May 2014, pp. 91–98, 2008.

[51] J. Meijer, Q. Lei, and M. Wisse, "An empirical study of single-query motion planning for grasp execution," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1234–1241, 2017.

[52] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, April 2013, pp. 1–6.

A

Demonstrations

During Demonstrations, the initial position of an object is obtained by the vision module (detailed in Appendix D). Then the haptic device is used to move the end-effector from a start pose to the desired pre-grasp pose. To conveniently provide the demonstrations using the 7-DOF manipulator of the care robot, the Geomagic Touch was used as haptic device. An illustration is depicted in Figure A.1. This device is used to control the end-effector position and orientation (6-DOF), and the Whole Body Controller (WBC) uses an optimization algorithm to determine the joint positions (7-DOF).

Figure A.1: Geomagic Touch device used to generate the demonstrations.

The end-effector poses in time $x(t)$ are recorded and saved in a database. There's an illustration of a demonstration in 2D in Figure A.2. This plot identifies two main issues. The demonstration contains many data points, and there is a lot of noise at the end of the motion. The noise is a problem for learning, because the model will have this noise as well. Whenever the robot uses a learned motion to generate a new motion, there will be a lot of unnecessary movements. Therefore, the demonstration needs to be filtered before learning a motion. The motion is filtered by hand in this case. However, a filter could be applied to get a better motion.

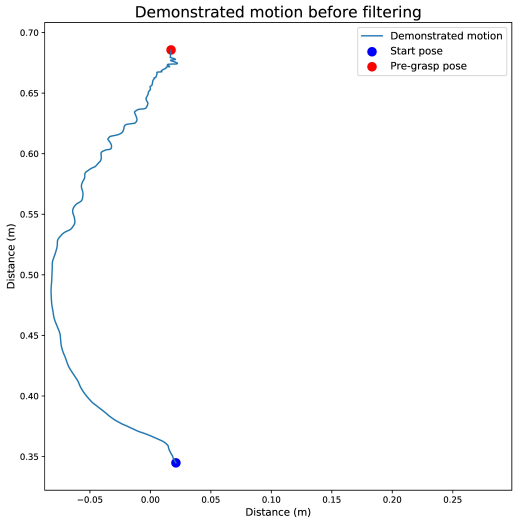
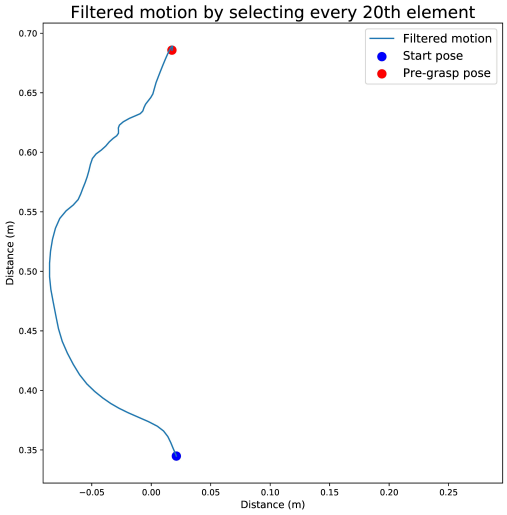
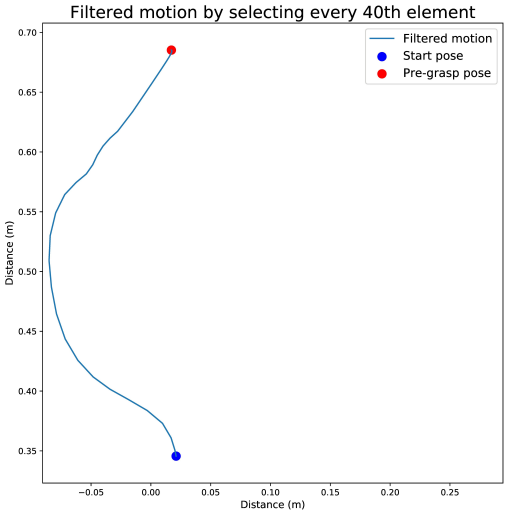


Figure A.2: The recorded end-effector position in 2D. The plot shows that there is a lot of noise at the end of the motion. Besides, a lot of data points are recorded. A filter should be applied before learning a motion in order to get a more legible motion.

In order to reduce the noise in the data, the amount of data points is reduced by selecting every nth element. As an example, see Figures A.3a and A.3b, where the 20th and 40th elements are selected, respectively. There is a trade-off between encoding unimportant motion and not encoding relevant information. Learning is accomplished by using motions with 15 to 20 data points. Figures A.4a and A.4b provide a 3D illustration of a demonstration and the motion after filtering.



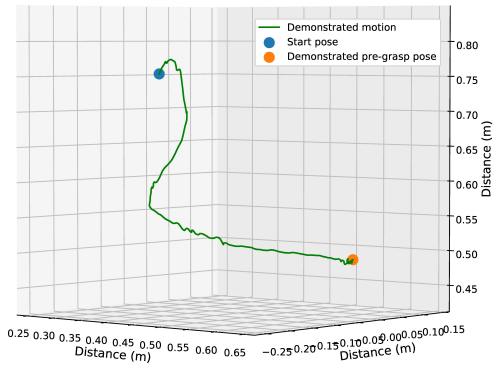
(a) Every 20th data point is selected which result in a smoother demonstration.



(b) Every 40th data point is selected which result in a smoother demonstration.

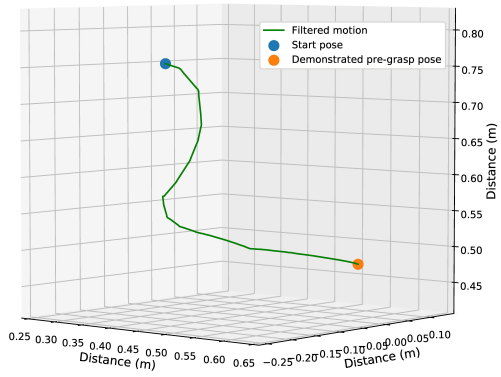
The data had a lot of end-effector vibrations. These vibrations can be caused by the operator or the robot's controller. We have seen that the controller causes some noise. However, controlling the robot is beyond the scope of this project. Analyzing this might enable better demonstrations.

A 3D motion before filtering



(a) Demonstration in 3D without filtering the motion.

A 3D motion after filtering



(b) Motion in 3D with the filtering.

B

Tools

B.1. Introduction to MoveIt

MoveIt is a state-of-the-art open source platform for mobile manipulators. The ROS interface includes inverse kinematic solvers, path planning algorithms, and collision detection algorithms. This thesis utilizes the move_group Python API to communicate with MoveIt. Figure B.1 presents a brief outline of the MoveIt framework. There are several MoveIt commands which can be used to perform tasks such as moving collision-free to a pose or modeling collision objects for the planning scene. The implementation of the motion planner will be briefly described in this Appendix. After that, the implementation of the LfD framework using MoveIt is discussed.

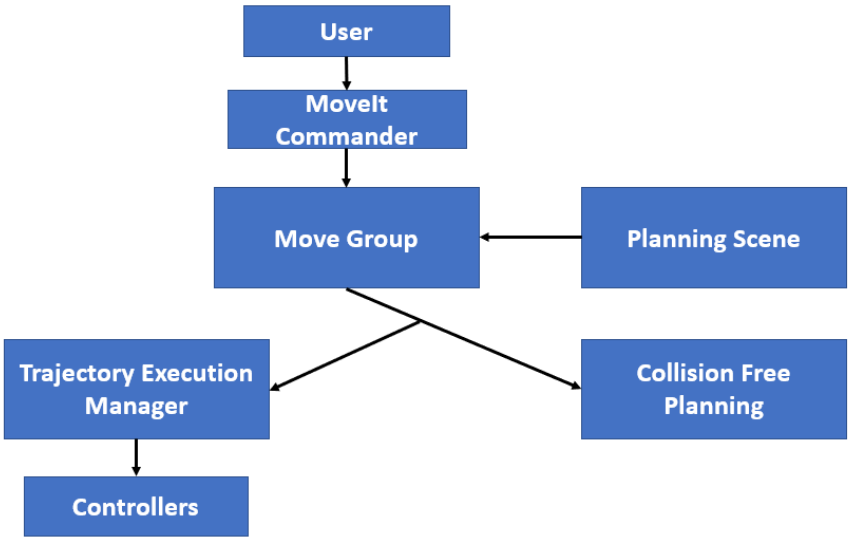


Figure B.1: The MoveIt pipeline is visualized in this figure. The user can give input to the Move Group by using the MoveIt Commander. The Move Group gets information of the Planning Scene. After that, a collision free plan can be made and executed.

B.2. MoveIt Motion Planning

B.2.1. Initializing

Detailed instructions are provided in a MoveIt tutorial on how to use the Move Group Python Interface [1]. The moveit_commander and rospy node are initialized first. Next, there is a RobotCommander object

[1] https://ros-planning.github.io/moveit_tutorials/doc/move_group_python_interface/move_group_python_interface_tutorial.html

that provides information about the robot's kinematics as well as the current joint state of the robot. After that, a PlanningSceneInterface object is required. A Planning Scene is used to build a world and inform the robot how the surrounding environment is structured. It can be used to ensure that the robot moves in a collision-free manner. The final step is to create a MoveGroupCommander object. By doing so, MoveIt is able to control the robot's joints.

B.2.2. Planning Scene

It is essential that the robot understands its surroundings before it can plan and execute a motion. Hence, the first step after initialization is to make the robot understand what the scene is. A table can be modeled by a solid primitive as a box with known dimensions and positions (in the base footprint frame):

Listing B.1: Modeling a Table

```
table = moveit_msgs.msg.CollisionObject()
table.id = "table"
table.header.frame_id = "base_footprint"
table.operation =  table.ADD

table_primitive = shape_msgs.msg.SolidPrimitive()
table_primitive.type = shape_msgs.msg.SolidPrimitive.BOX
table_primitive.dimensions = [1.0, 1.5, 0.8]
table_pose = Pose()
table_pose.position.x = 1.15
table_pose.position.y = 0.0
table_pose.position.z = 0.4

table.primitives = [table_primitive]
table.primitive_poses = [table_pose]
```

The cereal box is composed of information from the AruCo marker and known dimensions. Here, the pose from the vision module is transformed into the base footprint frame and then used as the position for the object. The z-position of the AruCo marker is the top of the box. To correct this, half of the height of the box should be subtracted in order to get the pose of the box. The final step is to add these objects to the Planning Scene by:

Listing B.2: Adding objects to Scene

```
self.scene.add_object(table)
self.scene.add_object(object_pick)
```

B.2.3. Planning and Execution

It is now possible to grasp and plan an object safely with MoveIt. Initially, the robot has to move into a predetermined start position. The robot can be moved into a defined pose using the following function:

Listing B.3: Move to a Pose

```
def go_to_pose_goal(self, pose_goal):
## Planning to a Pose Goal.
## The pose plan is the plan made by the MoveIt planner
    pose_target= self.move_group.set_pose_target(pose_goal)
    pose_plan= self.move_group.plan()

## This pose_plan can now be executed
    ex = self.move_group.execute(pose_plan, wait=True)
## Now we want to have no residual movement,
## therefore we use the function stop().
## The function clear make sure that the targets are removed.
    self.move_group.stop()
    self.move_group.clear_pose_targets()
```

The next step is to pick up the object with the MoveIt pick function. In this thesis, we follow the C++ approach described by MoveIt[2] and implemented that in Python for the TIAGo robot. In order to ensure that the object can be grasped in each position, the pre-grasp pose is set up to be a function of the box pose. As a result, it is not necessary to program always a grasping position for the robot. A pre-grasp and a grasp pose have to be determined before a pick can be performed. In a case with no orientation (see Figure B.2), the pre-grasp should have the same y coordinate as the box and the same x coordinate as the box minus a distance (26 cm in this case). Due to the fact that the object gets only a yaw angle, the z-coordinate is always the same for the grasp pose as the object's z-coordinate. The gripper should be oriented in a manner that will enable it to grasp this object. It is a -90 degree yaw angle for the TIAGo robot to grasp a box as illustrated in Figure B.2.

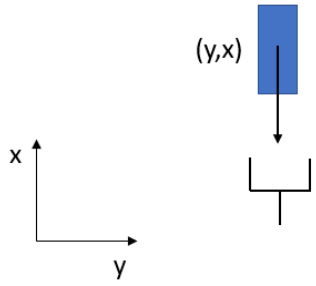


Figure B.2: Top view of the object. Grasping the cereal box that has no orientation. The gripper is in this figure at the pre-grasp pose. The end-effector has a -90 degrees yaw angle in this case.

The pre-grasp pose changes in situations where the object is rotated. An illustration can be found in Figure B.3. The vector in Figure B.2 gets the same orientation (alpha) as the object but still has the same length. The change in orientation of the box needs to be added to that of the gripper in order to be aligned with the object.

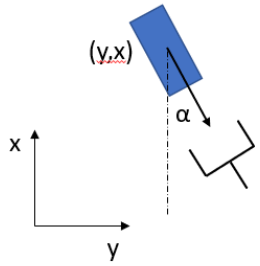


Figure B.3: Grasping the cereal box that has a orientation. The gripper is in this figure at the pre-grasp pose.

After making a grasp message which contains information about the pre-grasp and grasp pose of the object, the MoveIt pick function can be called to plan and execute a grasp:

Listing B.4: Grasp object

```
self.move_group.set_support_surface_name("table")
## It is also possible to only plan by setting plan_only to true.
pick_the_object= self.move_group.pick("object_pick", grasps, plan_only=False)
```

The set support surface is necessary to let MoveIt know that those objects are in a 'collision'. The function pick can pick the object named object_pick with the defined grasps.

[2] http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/pick_place/pick_place_tutorial.html

B.3. Combining MoveIt with LfD

By combining MoveIt with the LfD module the robot will be able to plan safe motions and execute them. Initializing and setting up the planning scene follow the same procedure as described in the previous section. Using the go to pose function, the robot can then move to the start pose.
The next step will be to develop a plan with the DMPs. MoveIt should provide the following information: the current end-effector pose, the pose of the object to be grasped, and the primitive to be used. The following functions return the current end-effector pose, using the arm_tool_link as the frame of the end-effector:

Listing B.5: Get end-effector pose

```
def end_effector_pose(self):
    self.move_group.set_end_effector_link("arm_tool_link")
    return self.move_group.get_current_pose()
```

The planning scene provides information about the grasped object. However, the z-coordinate should be converted to the marker position, since the pre-grasp pose is related to the marker position. As a result, it is necessary to add half a height of the box to the z-coordinate:

Listing B.6: Get end-effector pose

```
object_id = ["object_pick"]
visual_object_pose = self.scene.get_object_poses(object_id)
dim_box= self.scene.get_objects(["object_pick"])
dim_box_z= dim_box['object_pick'].primitives[0].dimensions[1]
visual_object_pose['object_pick'].position.z=
visual_object_pose['object_pick'].position.z + (dim_box_z/2)
```

It is possible to obtain the primitives for the DMP from either a State Machine or a Behavior Tree. A hard coded algorithm is used to determine which primitive should be loaded and used by the robot. A function that calls the DMP module receives the current end-effector pose, object pose, and motion primitive. This DMP function will calculate the pre-grasp pose of this situation based on the data from MoveIt and the database. A service request is then sent to the DMP module. A trajectory is published to the generated_dmp topic by the DMP module. The MoveIt script receives these messages and has to formulate a plan and execute it. In order to do this, MoveIt has to make a Cartesian plan from the waypoints that are provided by the message. When it is difficult to make a plan, the path is cut into several pieces. As a result, it is easier for the robot to develop plans and execute them. The core of the code is the following:

Listing B.7: Make Cartesian Plan

```
waypoints=[]
jump_threshold= 5.0
precision= 0.01
wpose = self.move_group.get_current_pose().pose
length = len(data.poses)

for i in range(0, length-1):
    wpose.position.x = data.poses[i].pose.position.x
    wpose.position.y = data.poses[i].pose.position.y
    wpose.position.z = data.poses[i].pose.position.z
    wpose.orientation.x = data.poses[i].pose.orientation.x
    wpose.orientation.y = data.poses[i].pose.orientation.y
    wpose.orientation.z = data.poses[i].pose.orientation.z
    wpose.orientation.w = data.poses[i].pose.orientation.w
    waypoints.append(copy.deepcopy(wpose))

(plan, fraction) = self.move_group.compute_cartesian_path(
    waypoints, precision, jump_threshold)
```

The data represents the path received from the DMPs. It is appended to a list called waypoints. After this, MoveIt can simply be used to follow those points with the compute_cartesian_path. The function returns a plan and the fraction of the waypoints it could follow. If this fraction is not 1, then a new plan should be developed by segmenting the list of waypoints. When this is not possible, another motion primitive should most likely be used. Another solution might be to reorient the arm. As for the precision, it is set to 0.01 m, which is how precisely the end-effector should pass through the points. Identifying the jump_threshold is crucial to the process. There is not enough documentation on this topic. In the first test, the threshold was set to zero. Unfortunately, the arm is not constrained when it comes in singularities. It is possible that the robot follows the waypoints and suddenly the arm needs to be reconfigured between waypoints. In the process of reorienting the arm, MoveIt does not check for self-collision with the robot. Figure B.4 shows the reproduced behaviour of the robot in simulation.

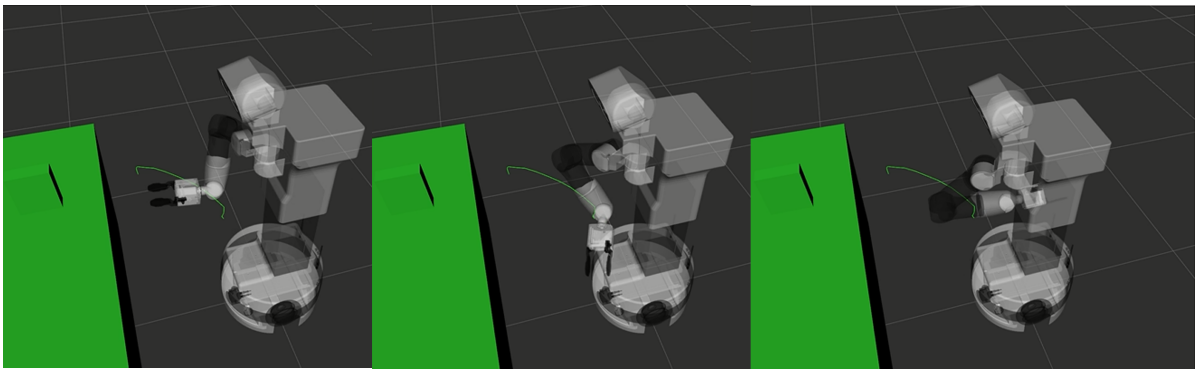


Figure B.4: Reproduced weird behaviour of the robot in simulation. The robot moves through the waypoints and suddenly changes the path to get a different configuration. In this process it could collide with the robot.

In order to prevent this behavior, the jump_threshold can be used as a ”constraint” on the robot. It limits the jumps between two waypoints in the joints. A value that is too low prevents the robot from making any plans. By trial and error, it was found that a value of 5.0 was successful for the TIAGo robot. The plan created by MoveIt is visualized in simulation. The robot can then be instructed to execute this plan:

Listing B.8: Execute plan

```
print("Press enter to generate new motion")
raw_input()
execution=self.execute_plan(plan
```

The last two steps involve grasping the object. This cannot be accomplished using the pick function. This is because MoveIt will not simply find a plan that moves in the shortest path to the object. Therefore, a grasp is made by following 4 waypoints towards the object. Another interesting possibility is to utilize a newly learned motion and execute it. To determine the four waypoints, we obtain the pose of the grasping frame of the robot and the edge of the cereal box. This is illustrated in Figure B.5.

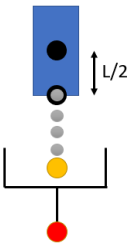


Figure B.5: Defining 4 waypoints between the gripper grasping frame and the edge of the box. The black dot is the pose of the cereal box, L is the length of the box, the yellow dot the grasping frame, and the gray dots the 4 waypoints.

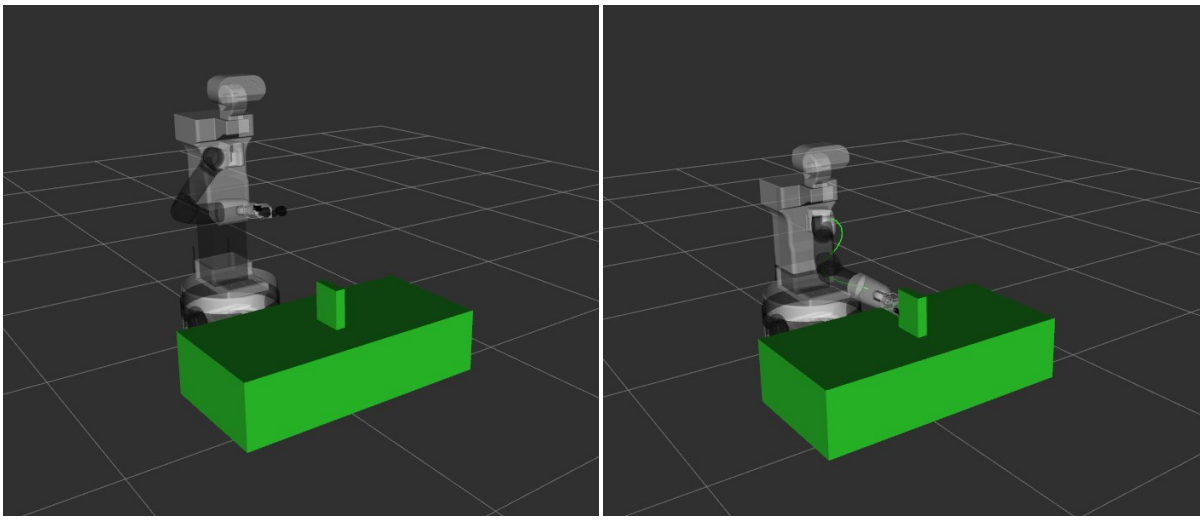
MoveIt can determine the pose of the grasping frame, which is represented by the yellow dot. The gripper frame is the same as the arm_tool_link frame (red dot), but translated inside the gripper. The

object's information is derived from the Planning Scene. After defining the waypoints, a plan can be developed and executed as described above.

C

Generalization in Simulation

In this part, several figures are shown of how the teleoperated LfD method generalizes. Figure C.1a shows the start pose of the end-effector before doing the demonstration and the place of the cereal box. Figure C.1b shows the executed trajectory of the learned motion without generalization. The cereal box is in the same pose as in the demonstrated data. The robot grasps the cereal box correctly with the same motion and from the same side.

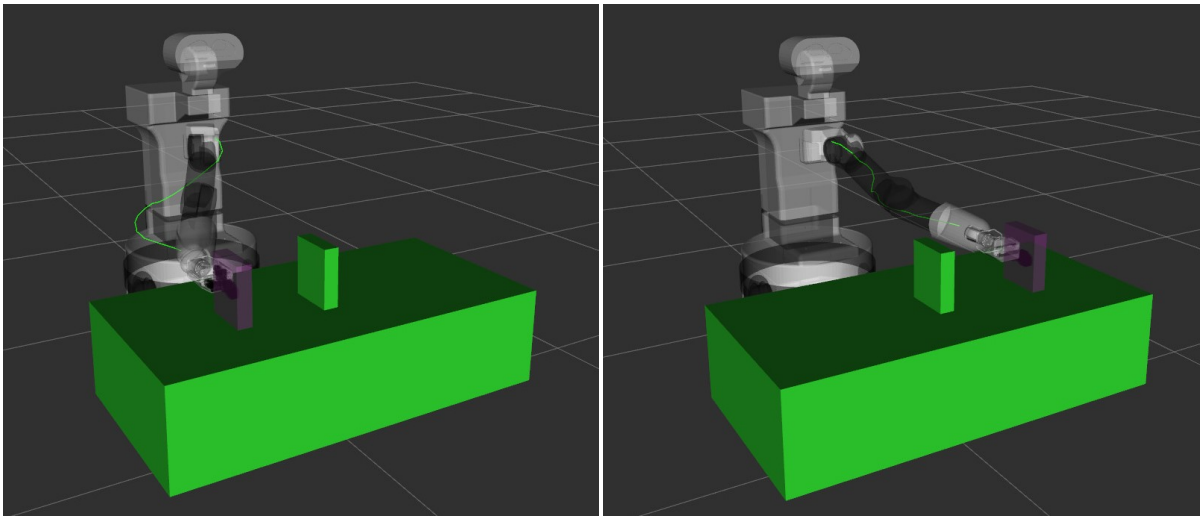


(a) Start pose of end-effector before executing a learned motion.

(b) Successfully grasped the cereal box.

Figure C.1: The operator demonstrated the grasp in the same position as this box. This figure shows that the robot was able to automatically grasp the object again from a certain start pose.

After this, the objects are placed more to the left or right of the robot. The robot starts with the same end-effector configuration as in Figure C.1a. The result of grasping those objects is shown in Figure C.2a and Figure C.2b. The green object is in all the figures the object with the same pose as used with the demonstration. The purple box is a successfully grasped cereal box. The only thing the robot needs to know is the pose of the cereal box. It generates autonomously a new grasp pose and a trajectory. After that, the robot executes the motion and grasps the object. The motion from Figure C.2a shows the same shape and grasp of the object. However, when the object is placed more to the left side of the robot the motion looks less like the learned motion. In that case, it would probably be better to use a new motion. A motion that is mirrored like in Figure C.3 could help in that case.



(a) Cereal box placed more to the right side of the robot.

(b) Cereal box placed more to the left side of the robot.

Figure C.2: Successfully grasped the cereal boxes that are placed more to the right or left side as demonstrated (from robot's perspective). The right figure shows a motion that is becoming strange for a human.

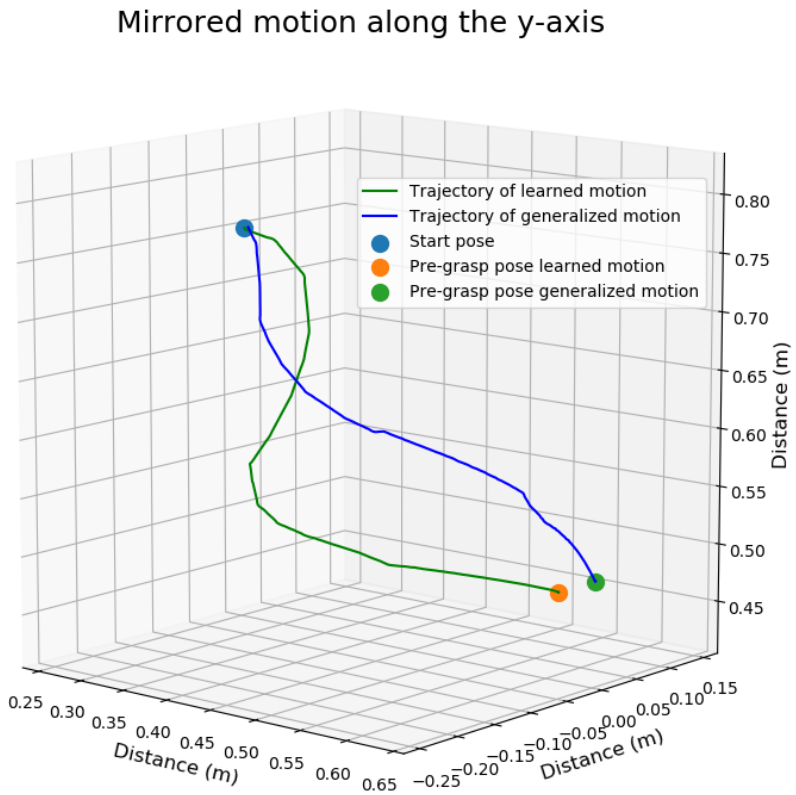
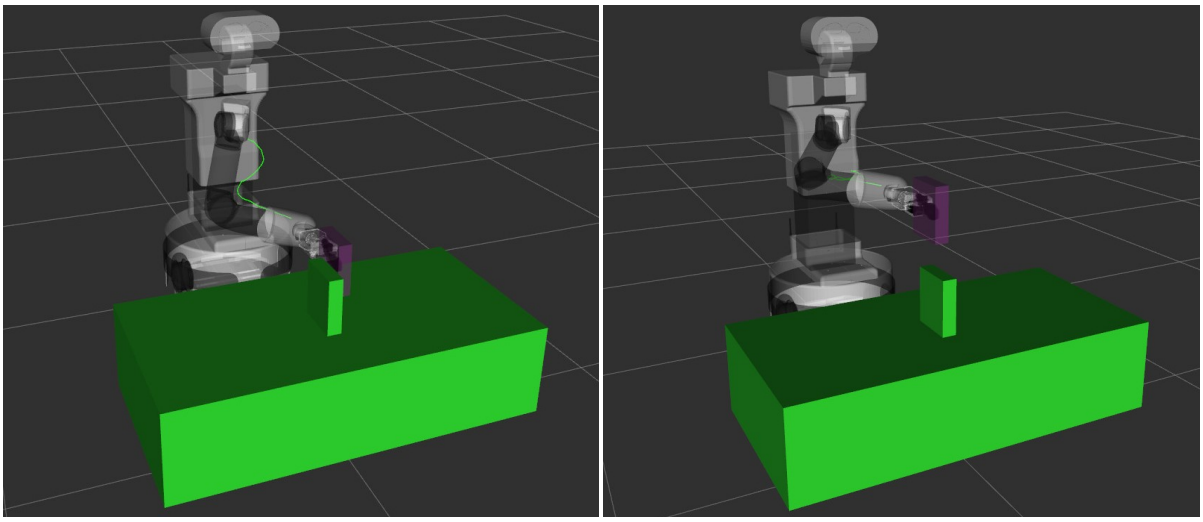


Figure C.3: A mirrored motion is shown in this figure. A motion like this could probably be used when a motion with exaggeration to another side is needed.

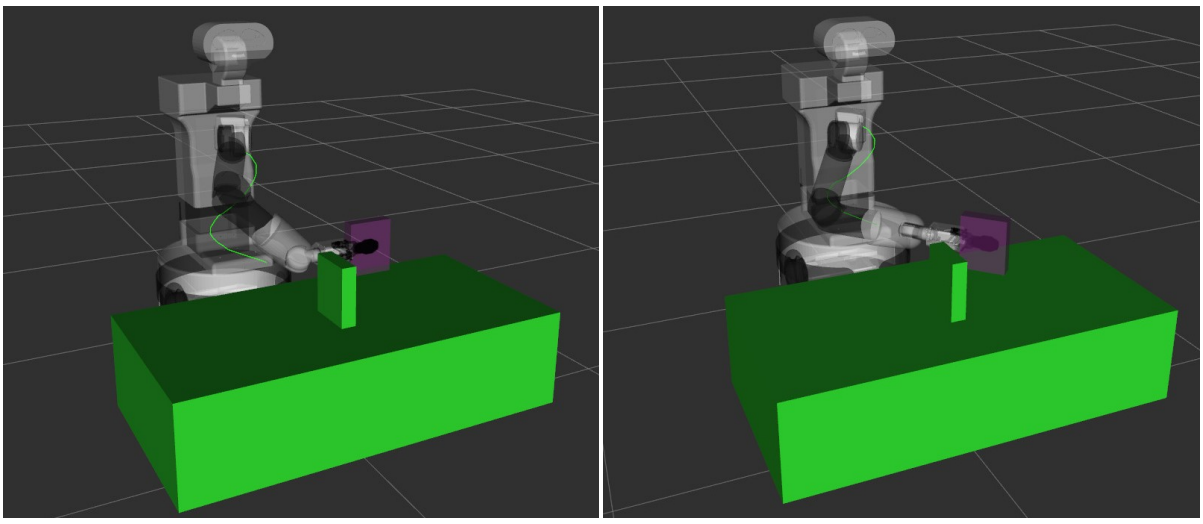
Another possibility is that the object is placed closer to the robot and also more to the left or right. In Figure C.4a the object was placed closer and more to the left of the robot. Again the object is well grasped with a learned motion. In Figure C.4b the object is placed higher than the object in the demonstration. Again the generation and execution of the grasp is successful.



(a) The cereal box is placed closer and more to the left of the robot. **(b)** The cereal box is placed higher than the demonstrated one.

Figure C.4: A cereal box that is translated in two different directions is shown in the left figure. The robot was still able to grasp it. The right figure shows a box that has is placed higher than the demonstrated one.

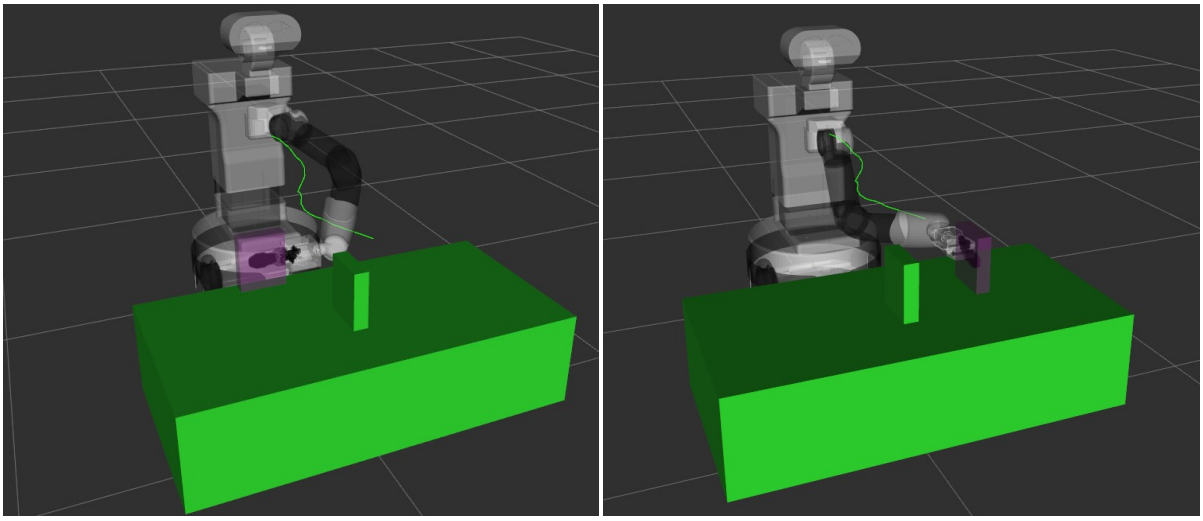
The previous examples showed objects that had the same orientation as in the demonstration. In the next two examples, the cereal box is placed with a different position and orientation. Figure C.5a shows a grasp where the cereal box is 90 degrees rotated over the yaw axis with respect of the object pose that was used during the demonstration. Figure C.5b shows a grasp of a cereal box that is 45 degrees rotated. Both figures show that the robot generates a learned grasp without problems.



(a) 90 degrees rotation over the yaw axis of the cereal box. **(b)** 45 degrees rotation over the yaw axis of the cereal box.

Figure C.5: Different positions and orientations of the cereal box. Both cases are successfully grasped. The LfD method is able to generalize to situations where both the object is translated and rotated.

However, when the object is -90 degrees rotated (Figure C.6a the motion becomes like the one shown in Figure C.2b. It can successfully grasp the cereal box in this case but again a different motion is preferred. Figure C.6b is again a cereal box that is randomly placed elsewhere. It is more to the left side and closer to the robot.

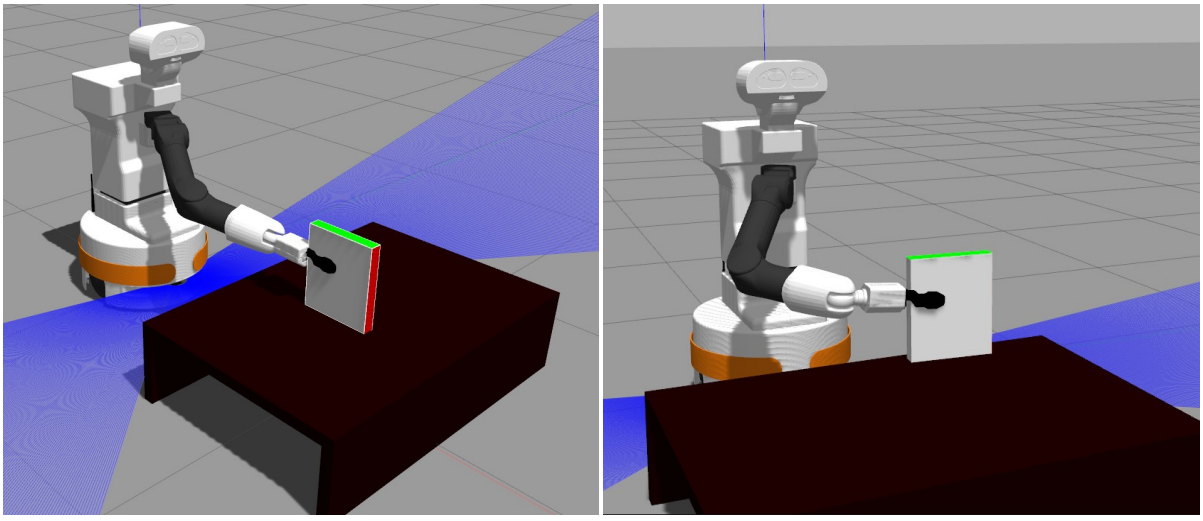


(a) -90 degrees rotation over the yaw axis of the cereal box.

(b) A random translated box.

Figure C.6: Different position and orientation of the cereal box. Both cases are successfully grasped. The left side is a strange motion for the human. However, the robot was still able to grasp it. The right side show a box with a random translation. In this case the motion is still useful but probably is becoming weird for a human.

The shown figures are RVIZ simulations which has no physics in the environment. However, in the background the Gazebo simulation, which is like the real robot with a table and cereal box, was running. The robot could grasp the cereal box also in that simulation as is shown in Figure C.7a and C.7b.



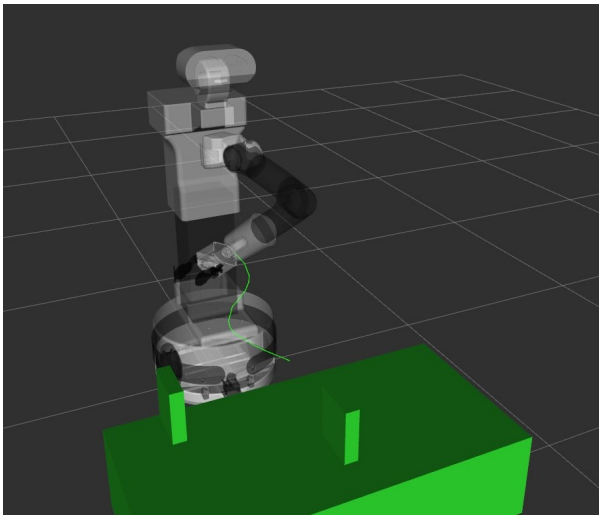
(a) Same object pose as used by the demonstration.

(b) 90 degrees rotation over the yaw axis of the cereal box.

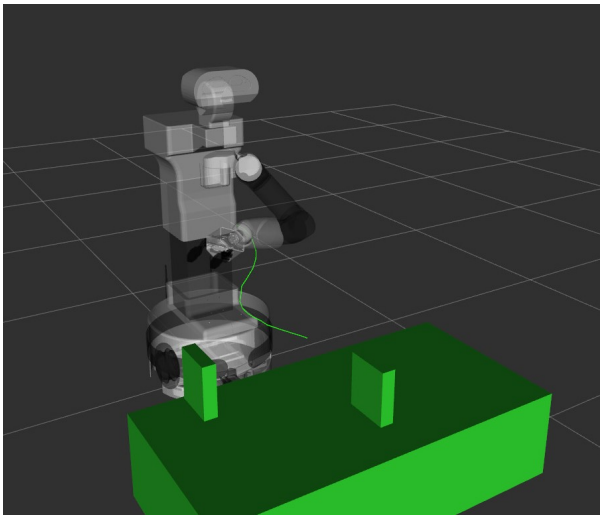
Figure C.7: Gazebo simulation with physics. The robot is able to grasp the box in different situations.

After this, we visually analyzed how different start poses are working. Most of the cases worked perfectly. However, one interesting result is that the roll rotation of the end-effector is important. Figure C.8a shows a different start pose than the one used in the demonstration. A motion is generated which can be successfully executed. However, Figure C.8b shows the same start position but with a different orientation of the end-effector. It looks almost the same but the end-effector is 180 degrees rotated over the roll axis. For an operator, it is hard to understand what is happening. It can generate a new trajectory but the robot cannot execute this motion due to this orientation. Figure C.9a and Figure C.9b show the same behavior, with the first one a working motion and the second one a trajectory that can not be executed. A solution could be an additional motion that changes the configuration of the end-

effector. Another solution might be to demonstrate that the object can also be grasped in a different orientation of the end-effector. In that case the robot could choose between grasps.

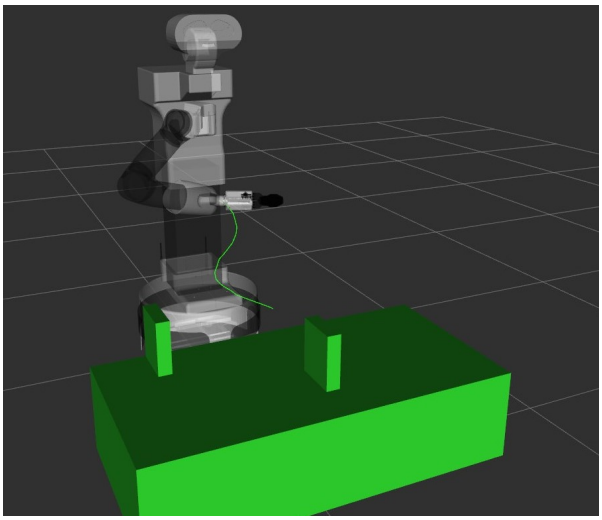


(a) Working grasp with different start pose. The end-effector is in a different orientation than demonstrated.

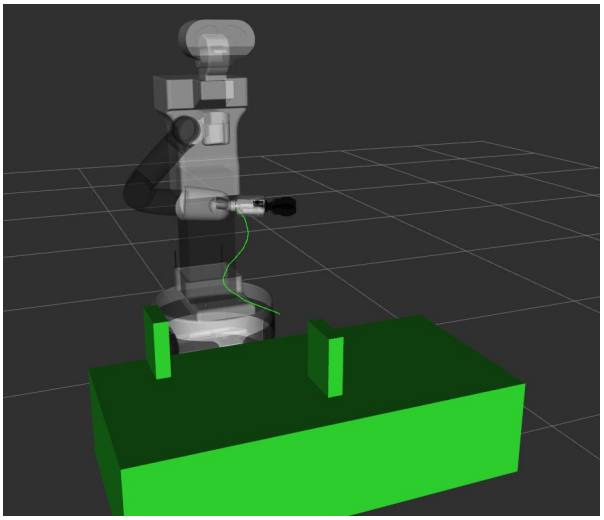


(b) Not working start pose because the end-effector is 180 degrees rotated over the roll axis with respect to the left figure.

Figure C.8: Different start poses used to grasp the cereal box. With the left side showing a start configuration of the end-effector what could be used to successfully grasp the object. The right side is almost the same, however the robot is not able to execute a motion in this case. It shows that the roll angle of the end-effector is important.



(a) Working grasp with different start pose. The end-effector is in a different orientation than demonstrated.



(b) Not working start pose because the end-effector is 180 degrees rotated over the roll axis with respect to the left figure.

Figure C.9: Different start poses used to grasp the cereal box. Again the robot is not able to grasp the object in the right figure.

D

Detection of Object Pose

For the experiment, a vision module is implemented in order to get the poses of the objects. By using a vision module it can be seen what small noise in vision can do to the learning module. A vision module is not within the scope of this project. Nevertheless, I wanted to combine a vision module with the LfD approach to see how it can automatically grasp objects. The conditions for the module is that it can detect a pose (position and orientation) of the object and that it is not too time consuming to make it work. First of all, texture detection is used to get a pose of a texture on a surface. It was difficult for the robot to detect textures on smaller objects. Therefore, color detection was implemented after that. The detection module did present some issues which meant that it was time consuming to fix them. As a result, ArUcO detection is used to get the pose of an object. The methods will be discussed shortly, along with the issues.

D.1. Texture Detection

Firstly, texture detection is used to detect a box that has an image on top of it. Texture detection is based on feature matching between the camera input and a reference image of the texture. Then, the pose of the texture, and thus of the object, is determined by homography estimation and the size of the texture. The implementation is based on tutorials of feature detection[1] and planar texture detection[2]. A large poster (0.5m by 0.64m) of a texture could be detected in simulation. This poster is scaled and placed as a texture on a box with the top surface of 0.1m by 0.13m. Figure D.1 shows the Gazebo simulation with a box that has this texture on it. A pose must be detected on this box in order to use

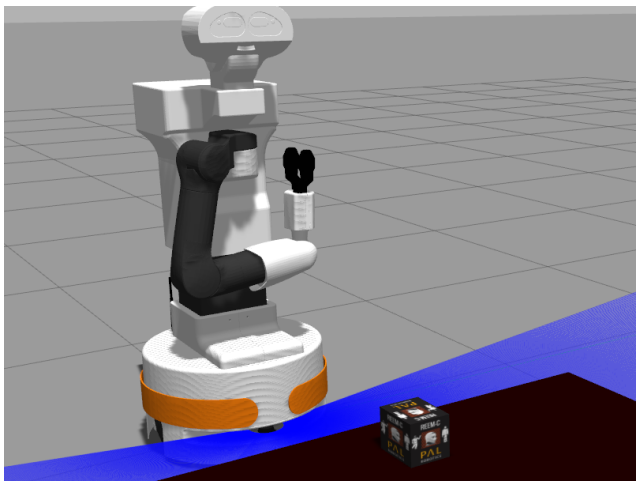


Figure D.1: A box with a texture on it for texture detection. The robot will try to detect the box pose by detecting the texture.

[1] `http://wiki.ros.org/Robots/TIAGo/Tutorials/Matching`
[2] `{}http://wiki.ros.org/Robots/TIAGo/Tutorials/HomographyEstimation`

it in the LfD module. An image of the texture and the size of the top size of the box are provided to the module. If the texture is detected, a pose will be returned from the middle of the image. Image D.2 shows in RVIZ the texture pose for the box when it is rotated 90 degrees (top of box facing the robot). The detection module can accurately identify the box by detecting its texture. However, in a lot of poses, the box cannot be detected because the texture is too small. Therefore, we could not use this method to detect the object pose.

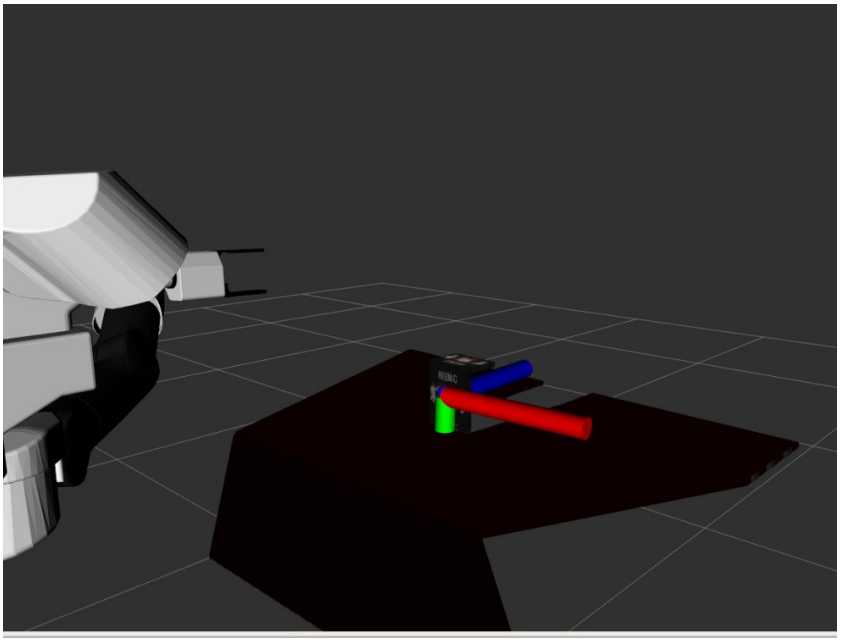


Figure D.2: A pose is returned after the texture has been detected. The pose is visualized using an axes. In this case the robot was able to detect the texture.

D.2. Color Detection

Instead of adding a texture on top of the box it is also possible to color the top side of the box and use color detection from OpenCV[3]. Therefore, a box is made in simulation with a green top side (see Figure D.3). In the module the green color will be detected in the RGB camera image. A rectangle is drawn around the green surface and a 2D position of the middle is returned. With the depth camera a 3D position is determined in the camera frame. After that, this position is transformed to the base frame. This approach works fine for determining a 3D position. However, the orientation of the object is also important. By using PCA the orientation can be calculated of the object. However, for an easy implementation it is better to have a camera that is perpendicular to the top side of the cereal box. Otherwise the object gets already an orientation by only placing it to the left or right. This problem is shown in Figure D.3 at the top right side. The cereal box is placed to the left of the robot and has no orientation. The robot, however, sees the object with an orientation. Two solutions are suggested, namely using a different camera for object detection that is placed perpendicular to the objects or to use homography. Another camera is not preferred because we want to use the robot only. Homography is too time consuming. Therefore, ArUco marker detection is tried in the next section.

D.3. ArUco Detection

With the TIAGo robot it is possible to use ArUco marker detection[4]. An ArUco cube was used in a pick-and-place pipeline[5]. After some changes, the ArUco module could be used for our TIAGo robot. The result is that the ArUco cube (which is shown in Figure D.4a) in the Gazebo simulation could be

[3] `https://opencv.org/`
[4] `http://wiki.ros.org/Robots/TIAGo/Tutorials/ArucoBoard`
[5] `http://wiki.ros.org/Robots/TIAGo/Tutorials/MoveIt/Pick_place`

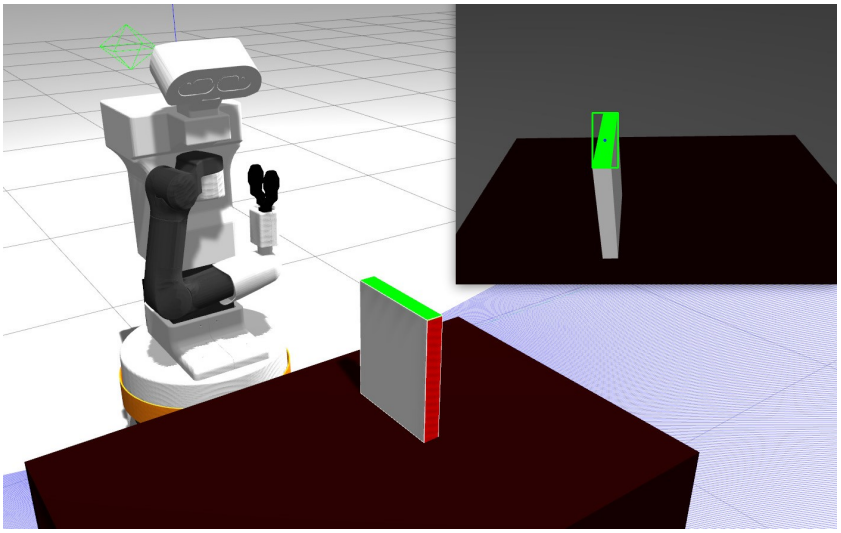
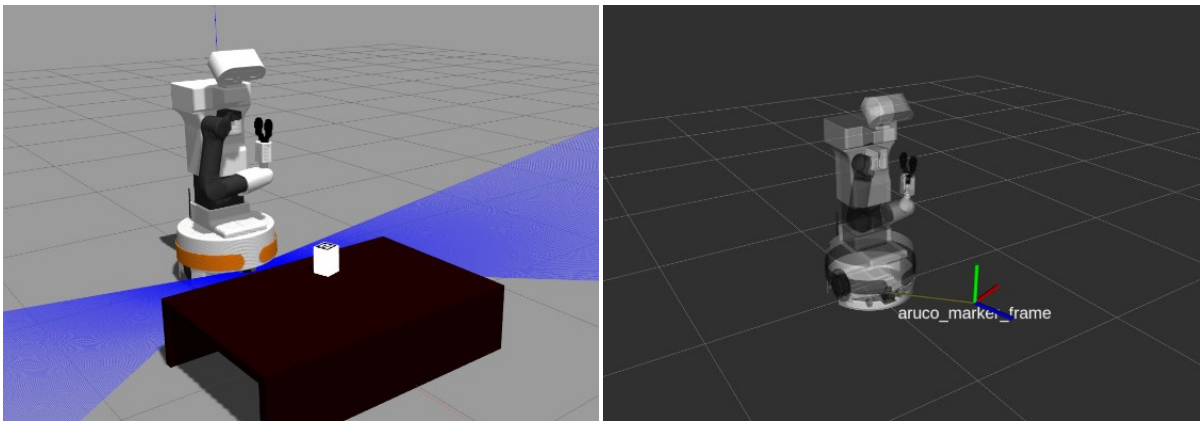


Figure D.3: Problem with color detection. The object is placed more to the left side of the robot without having a orientation. However, the robot sees the object as it has an orientation.

detected. A pose is published, which is visualized in Figure D.4b. This is used to detect a cereal box with a marker on top of it (Figure D.5). Hereafter, a separate camera is used (same as the robot camera) to see whether a real camera could detect the marker. Figure D.6 shows that the camera was able to detect the marker in reality.



(a) ArUco cube in Gazebo simulation for detection.

(b) Aruco marker is detected and a frame is published in RViz.

Figure D.4: The left figure shows the Gazebo simulation with the robot and a table with an ArUco cube on top of it. The right figure shows that the robot is able to detect it. A frame is published which corresponds with the pose of the marker.

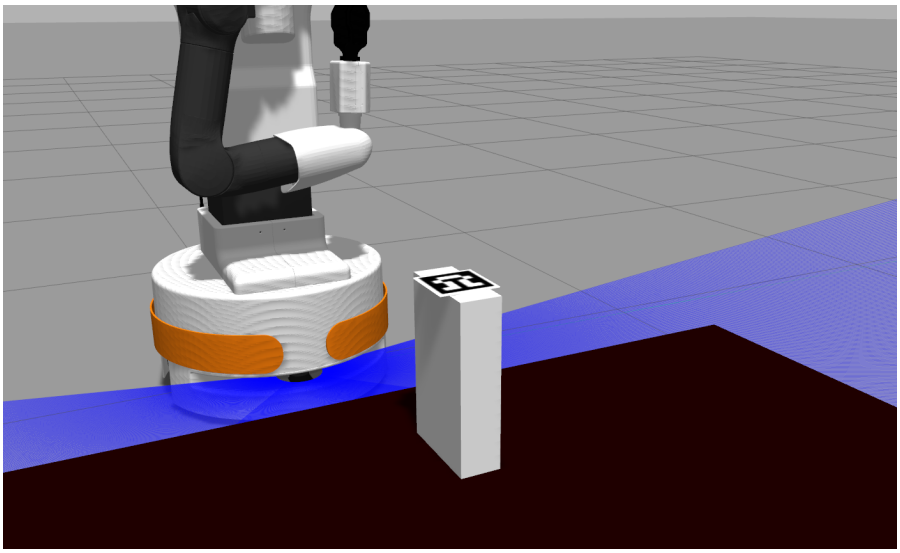


Figure D.5: ArUco marker placed on a cereal box in simulation.

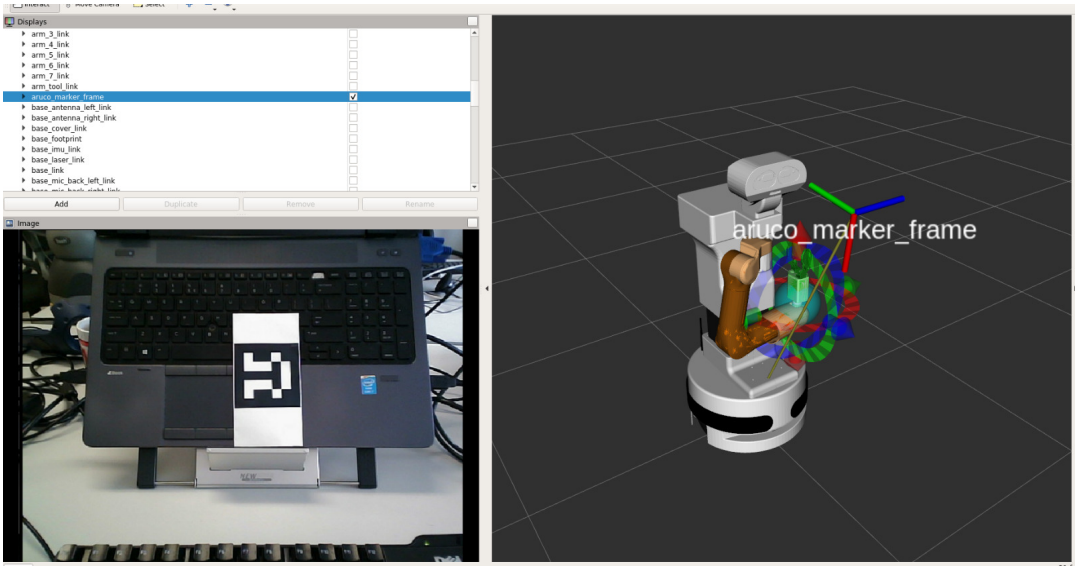


Figure D.6: ArUco marker is detected with a real camera.

E

Additional Results of Experiment

In Table E.1 and Table E.2 the data of the motions shown in the video of the end-effector and elbow are presented.

Table E.1: End-effector data of the experiment with participants. The Jerk is the integrated square jerk, cross are the zero-crossings of the velocity. All data is calculated from the TF data obtained from the robot.

Place video	Sort	V_mean (m/s)	V_max (m/s)	A_mean (m/s^2)	A_max (m/s^2)	Jerk (m^2/s^5)	Cross	Path length (m)
1	LfD 4	0,035	0,176	0,234	3,390	2375	5	0,622
2	MP 1	0,048	0,311	0,294	5,017	7481	4	0,983
3	MP 6	0,057	0,594	0,227	4,296	8922	13	2,350
4	LfD 7	0,035	0,181	0,251	3,068	4410	2	0,706
5	MP 4	0,082	0,508	0,397	10,874	18813	9	1,995
6	LfD 2	0,030	0,155	0,182	2,693	2144	1	0,588
7	MP 5	0,079	0,779	0,602	10,675	78936	6	1,924
8	MP 7	0,090	0,782	0,640	26,763	70088	9	2,574
9	LfD 3	0,025	0,192	0,166	3,141	2677	3	0,724
10	LfD 1	0,021	0,235	0,141	3,750	4817	1	0,655
11	LfD 5	0,032	0,171	0,198	2,752	2609	3	0,699
12	MP 2	0,028	0,253	0,144	4,720	7640	5	0,852
13	MP 4	0,082	0,508	0,397	10,874	18813	9	1,995
14	MP 3	0,041	0,382	0,164	6,806	5239	9	1,352
15	LfD 6	0,035	0,182	0,232	3,081	4589	1	0,664
16	LfD 4	0,035	0,176	0,234	3,390	2375	5	0,622

Table E.2: Elbow data of the experiment with participants. The Jerk is the integrated square jerk, cross are the zero-crossings of the velocity. All data is calculated from the TF data obtained from the robot.

Place video	Sort	V_mean (m/s)	V_max (m/s)	A_mean (m/s^2)	A_max (m/s^2)	Jerk (m^2/s^5)	Cross	Path length (m)
1	LfD 4	0,029	0,160	0,185	2,852	1663	4	0,511
2	MP 1	0,036	0,234	0,220	3,793	4018	7	0,739
3	MP 6	0,031	0,220	0,118	1,835	2379	7	1,292
4	LfD 7	0,032	0,171	0,220	3,066	3532	6	0,637
5	MP 4	0,046	0,312	0,209	3,028	3923	6	1,119
6	LfD 2	0,026	0,135	0,156	2,347	1652	7	0,510
7	MP 5	0,042	0,381	0,312	5,926	18086	5	1,030
8	MP 7	0,044	0,361	0,286	6,390	13711	7	1,266
9	LfD 3	0,021	0,174	0,124	2,234	1439	5	0,595
10	LfD 1	0,017	0,206	0,116	3,232	3705	7	0,543
11	LfD 5	0,026	0,183	0,151	2,565	1731	3	0,551
12	MP 2	0,028	0,307	0,134	4,903	9765	6	0,855
13	MP 4	0,046	0,312	0,209	3,028	3923	6	1,119
14	MP3	0,027	0,267	0,102	3,872	2335	5	0,875
15	LfD 6	0,026	0,137	0,165	2,644	2131	3	0,493
16	LfD 4	0,029	0,160	0,185	2,852	1663	4	0,511

F

Instructions to Participants of Legibility Experiment

Experiment intent-expressiveness of TIAGo robot

Researcher: M.H. van Beem

May, 2022

Introduction

Your participation in this experiment is greatly appreciated. The purpose of the experiment is to measure the legibility of the robot's movement. In other words, can the robot's intention be deduced from observing a small part of its motion. In this experiment, you will make predictions about what the robot will do. Videos will illustrate these movements. First, some paperwork. After that, the experiment will be further explained.

Important Notes

- **At any time during the study, before, during, or after it, you are free to withdraw from participation.**
- **There is no restriction on the questions you may ask as long as they do not influence the study's outcome.**
- **There is no obligation for you to respond to any of the researcher's questions.**
- **We will use your information only to conduct academic research and pseudo anonymize it.**
- **Your confidentiality will be protected both internally and externally, and only the researcher will have access to it.**
- **You can get in touch with the researcher in question about your participation or the results of the study by contacting: m.h.vanbeem@student.tudelft.nl**

Form of Informed Consent

Taking part in the study

It has been read to me or I have read the above important notes of this study dated May, 2022. It was possible for me to ask questions about the study, and the questions I asked were answered satisfactorily.

◯ Yes

◯ No

By participating in this study, I agree to be a volunteer and understand that I may refuse to answer any of the researcher's questions. I am aware that I can withdraw from the study at any time, without having to provide a reason.

◯ Yes

◯ No

Privacy

Taking part in the study will expose my personal information to the researcher. To safeguard my identity, I know my personal information will not be published publicly.

◯ Yes

◯ No

This research may involve sensitive information about my own identity, which, if published, will remain anonymous. I understand that my information will only be used for academic research purposes.

◯ Yes

◯ No

I agree to have my pseudo anonymized research data from this particular study archived in the TU Delft repository for future research and learning purposes.

◯ Yes

◯ No

Name Participant:

Date:

Experience with robot motion planning:

◯ None ◯ 0-3 months ◯ 3-6 months

◯ 6-12 months ◯ 1-2 years ◯ More than 2 years

Experiment

In this experiment you will see a movie that consists of shorter videos of the robot. Those short videos are featuring the robot and two identical cereal boxes. One of the boxes will be grabbed by the robot. However, you will only see a few seconds of the movement and you have to predict whether the **left** or **right** cereal box will be grasped (from **your** perspective). The video will then be interrupted by a image with a 15 seconds timer so you can fill in the answer. In addition, you have to state how certain you are about your prediction. You may pause the video if you need extra time to fill in the form.

The first two motions showed in the video will be test videos to get familiar with the experiment and the form. However, the video will continue playing. So if you have questions after the test videos please pause the video and ask those. Now let's start the video!

Test video 1:

Predict the robot's actions based on your analysis of the video.

○ LEFT ○ RIGHT

How certain are you?

Uncertain ○ ○ ○ ○ ○ Certain

Test video 2:

Predict the robot's actions based on your analysis of the video.

○ LEFT ○ RIGHT

How certain are you?

Uncertain ○ ○ ○ ○ ○ Certain

Now you have finished the test videos. If you have questions please let me know!

Otherwise, you may continue to the next page.

Video 1:

1.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

1.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 2:

2.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

2.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 3:

3.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

3.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 4:

4.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

4.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 5:

5.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

5.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 6:

6.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

6.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 7:

7.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

7.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 8:

8.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

8.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 9:

9.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

9.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 10:

10.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

10.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 11:

11.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

11.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 12:

12.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

12.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 13:

13.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

13.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 14:

14.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

14.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 15:

15.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

15.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Video 16:

16.1 *Predict the robot's actions based on your analysis of the video.*

◯ LEFT ◯ RIGHT

16.2 *How certain are you?*

Uncertain ◯ ◯ ◯ ◯ ◯ Certain

Feedback:

Thank you for participating in this experiment. If there are any import issues to mention, please let me know:

Glossary

ADLs Activities of Daily Living i, 2

DMP Dynamical Movement Primitive 4, 5, 6, 21, 22

HIT Heemskerk Innovative Technology i, 2

LfD Learning from Demonstration i, ii, 2, 3, 4, 7, 8-12, 18, 21-24, 26, 29, 30, 33, 34

ProMPs Probabilistic Movement Primitives 4

WBC Whole Body Controller 4, 15