Programming by Demonstration: How to program Robots via Gestures?

Sarah Claudia Krings

Paderborn University, Paderborn, Germany skrings@campus.uni-paderborn.de

Abstract. To use robots efficiently, they have to be appropriately programmed and maintained. This has to be done by professionals on the topic and is often a very complex and time-consuming task, which thereby also makes it an expensive task. A way to improve this situation is Robot Programming by Demonstration, where the robot learns to solve new tasks not by being explicitly programmed, but by having the solution process of the task shown to it and building the skills from observing the demonstration. In this paper, the foundations of Robot Programming by Demonstration will be discussed and an overview over the literature on this topic will be given.

Keywords: Robot Programming, Programming by Demonstration, Imitation Learning

1 Introduction

Robots are becoming increasingly relevant in our society. They enhance production speed and precision, accomplish tasks which would endanger humans and are even used as toys. Especially in industrial contexts, robots are being used for very specific tasks which they sometimes fulfill over and over for a very long time. On the other hand, robots sometimes have to acquire new skills and functions. This can be the case if the robot is used for the first time or if it is used to fulfill a new task. Sometimes, robots are even expected to adjust their behaviour themselves to meet unforeseen circumstances, especially if they get into direct contact with humans.

All of this requires advanced programs which have to be fitted to each type of robot and each task for this robot. Programs for this have to be developed, tested, and kept up to date for the robot to work smoothly. Doing so, however, requires not only a high level of expertise in programming and robotics, but also takes a lot of time and effort. This can pose a big cost factor, which can even keep smaller companies from using robots at all, even if they are an effective solution. Furthermore, even not looking at the financial point of view, the need for extensive programming leads to over-complicated processes and slows down progress and new approaches. This reflects as well in the central goals of the VARobot project, which poses the research question of how to ease the task of robot operation and programming through novel interaction techniques.

Robot Programming by Demonstration (PbD) offers a solution to this problem: instead of lengthily programming robots "by hand", they can now learn procedures from a so called *teacher*. The teacher demonstrates how the new procedure is performed and the robot gains an "understanding" of how to complete the task from this. The demonstration as well as the observation of it can be done in different ways, which will be presented in depth later on.
A short example to introduce more context would be trying to teach a robot how to change the tires of a car. Firstly, the robot would be shown how to do this process, either by doing it itself while being controlled by the teacher, for example directly or using gestures, or watching the teacher exchange the tires. Depending on the concrete implementation, this demonstration might have to be repeated several times. From this information, the robot extracts how to perform the task (change tires) and can even generalize it to adjust to different circumstances (e.g. changing back tire instead of front tire).

According to Calinon et al. [6], PbD became a subject of interest in the beginning of 1980, where it was considered by factories using robots. The reasons were the ones mentioned above: reduction of the amount of programming work and therefore also a reduction of costs. In the beginning, the "demonstration" part consisted of manually teleoperating the robot to make it do the motions it was supposed to learn. In [10] for example, in addition to the recorded teleoperated movements, information about the shape and position of the target object and the surroundings were manually added due to the lack of sufficient image recognition capabilities. The learned information was then translated to rules from which actions could be derived again. In 1994, Münch et al. proposed to use machine learning, for example to find a structure to organize the detected kinds of operations [9]. A lot of the current work is fundamentally following similar approaches, but big changes have been made in the technical opportunities. Due to the technical progress, there are many new ways of demonstration, such as, for example, doing the demonstrations in virtual reality (VR) (see Figure 2) or guiding the robot directly by manually moving it (*kinesthetic guiding*) like in Figure 1. It is also possible to have a human teacher do the demonstrations and being monitored through cameras or motion sensors (see Figure 3) or even having sensor gear on the person.

As it is hinted here, the field of possibilities in Robot PbD is very wide. Furthermore, it has many facets and topics to go into further, such as precision, means of environment perception, and the capabilities of different demonstration sub-methods like gestures. For example, precision could depends on a number of other factors again, such as the precision of the robot's movements themselves and the quality of the learned movements, which in turn could depend on the quality of the demonstration, the recording of it, and the policy derivation from it. Due to the time and length constraints on this paper, it will not be possible to go into all of them in great detail, but to use the Robot PbD properly, a base of knowledge about the different tactics and details in programming by demonstration is necessary. To provide this base, a literature survey was conducted and will be presented in this work.

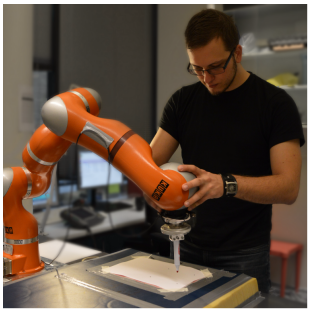

Fig. 1. A person kinesthetically guiding a robot through a task (from [14])

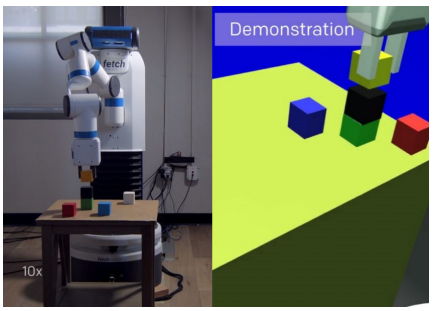

Fig. 2. Stacking of blocks being demonstrated in VR (left). The robot learns to stack real blocks from the virtual demonstration (right) (from [7])

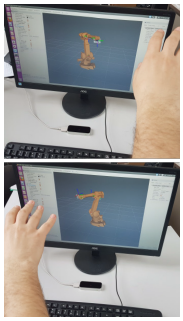

Fig. 3. A motion sensor detecting hand motions (from [8])

The remainder of the paper will therefore be structured as follows: In Section 2, the foundations of PbD will be treated and an overview over the different types of correspondence in demonstration and policy derivation will be given. Section 3 will provide more detail on the methodology of the literature survey. In Section 4, related work on the topic will be reviewed, sorted into the categories presented in Section 2 and compared on the different requirements from Section 3. The results of this will be discussed in Section 5.

2 Foundations

To introduce robot programming by demonstration, an overview of the topic will be presented. It will mainly be orientated on [5], as this work itself gives a very detailed overview and categorisation of robot PbD approaches and introduces categories and a structure in which these approaches can be sorted.
The works [6] and [15] will also be taken into account, but [5] was chosen because it has the most citations out of these works, indicating its relevance. Furthermore, the structure presented there seemed more general and universal than [15], which focuses on PbD for robotic assembly. It is easier to understand (especially for beginners on the topic of robot PbD and robotics) than [6], as this approach treats some areas quite theoretically or on a high level of abstraction.

2.1 Correspondence of teacher and learner

When a task is demonstrated to the robot, correspondence describes how the demonstration is related to the actions expected from the robot. [5] proposes an extensive categorisation with two main steps for this: *Record Mapping* and *Embodiment Mapping*.

Record mapping describes how the information about the task demonstration is recorded by the learner (the robot). It can be transferred directly and exactly as is was recorded (then the mapping is the identity function) or it can be encoded in some record mapping function (which is not the identity function). Here, it has to be mentioned that recorded data does not always only focus the teacher, the environmental state can be considered as well.
Embodiment mapping treats the way the information is recorded in the record mapping step is transferred to the learner for execution. Again, the recorded actions can either be executed exactly as given (mapped with identity function) or they can be put through a mapping (not the identity) before execution.

The two different possibilities in the embodiment mapping can be used to create two categories for the correspondence of approaches: *Demonstration* and *Imitation*. These categories and their combinations with different types of embodiment mapping will be explained and described in the following and are illustrated in Figure 4 as a matrix and in Figure 5 as a tree.

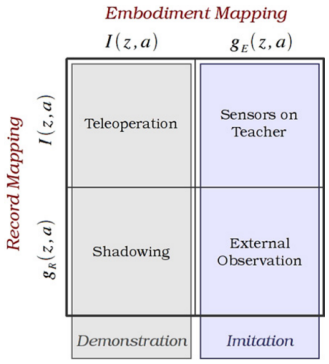

Fig. 4. The different combinations of record and embodiment mapping (taken from [5])

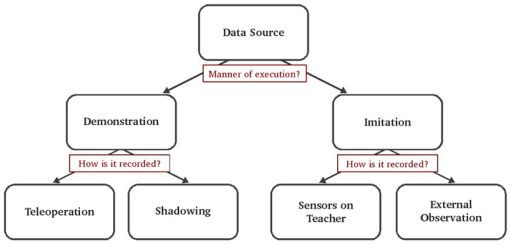

Fig. 5. A graph showing the connections between the different combinations of record and embodiment mapping (taken from [5])

2.1.1 Demonstration

In the category *Demonstration*, the embodiment mapping is the identity function, meaning the recorded information is directly reproduced. This works by performing the task demonstration directly on the robot learner itself, so all recorded actions are already in the correct form for execution and do not have to be adjusted to work on the robot.

Embodiment mapping which falls into the demonstration category can be combined with each of the approaches on record mapping.
Combining demonstration embodiment mapping with direct record mapping (record mapping with the identity as mapping function) can be done by *teleop-*

eration.
In teleoperation, the robot is operated by the teacher to perform the task it is supposed to learn. It records its own states and actions and can then repeat them later. Like mentioned before, no mappings are needed in this approach neither for recording nor for reproducing actions, because everything is done directly on the robot. Executing actions directly on the robot learner can be done different ways. One way that might come to mind first is controlling the robot through some kind of remote control like a joystick or similar appliances, which is used in several approaches [5], but can not accommodate for some situations, for example if a robot with complex motor controls should learn low-level skills. Controlling the robot via (body) gestures can face the same problem of lack in precision in the demonstration. Other techniques for teleoperation are for example controlling the robot via voice commands or using kinesthetic teaching, a method where the robot's joints are physically moved by the teacher to execute the wanted actions (this can also be seen in Figure 1). This does not work with all robots though, as it has to be possible to move the robot's joints "externally", so the robot has to make this possible and can not be stiff.

When demonstration embodiment mapping is combined with record mapping which encodes the passed information using a mapping function, the combination is called *shadowing*. When a robot learns by shadowing, it tries to mimic the motion the teacher demonstrates (mapping the teachers actions to itself) and records its own actions during the execution directly. To accomplish the mimicking, an algorithm (the mapping function) is needed to track and shadow the teacher.

2.1.2 Imitation

The other category of correspondence approaches is **Imitation**, where an embodiment mapping is needed to to convert the recorded information into a form which is usable for execution on the robot. This is necessary if the demonstration was performed on a platform different from the robot learner.

Similar to demonstration embodiment mapping, imitation embodiment mapping can be combined with either direct record mapping using the identity function or using a mapping function to encode the recorded information.

There is the *sensors-on-teacher* approach of equipping the teacher with sensors to directly record the teacher's execution. The recording therefore works without a mapping function, while (as it is an imitation technique) the data has to be put through an embodiment mapping function to make it usable on the robot learner.
The sensors-on-teacher method does remove the need for a record mapping function, however it requires quite specialized sensors, such as for example sensor suits for humans or rooms set up with sensors and cameras, to be able to measure the actions of the teacher. Later, some examples which use this approach will be presented, for example [2], where a data glove is used as a sensor.

Another approach is the *external observation* of the teacher. For this, the teacher is observed using external sensors which can, but do not have to, be the

robot learner's sensors. As these sensors are not directly on the teacher, a record mapping is needed to convert the collected data, and, as the demonstration is done by a teacher and not on the robot, an embodiment mapping is needed as well to make the data usable on the robot learner. Some external observation approaches, however, even map the recorded data directly to the robot. In comparison to the sensors-on-teacher approach, external observation records less precise data, but in return offers more generality as the specialized sensors from the sensors-on-teacher approach are not needed here. The external sensors used in external observation are often vision-based. For example, motion capture can be used to track human teachers.

2.2 Policy Derivation

Once example data for learning has been collected, policies have to be developed from it. This can be done in different ways. [5] proposes a categorisation into three main categories. The possibilities they present are either using a *mapping function*, a *system model*, or a *plan*, which will all presented in the following part. Figure 6 illustrates the different categories and their subcategories.

2.2.1 Mapping Function

A mapping function derives policies by using the collected data to create a mapping function which connects the robot's observations, about itself, but also about its context and environment, to actions it should execute in response to these.
The mapping function tries to approximate the function on which the teacher's actions are based (but which is not known to the learner). The training data from the learning process is also used to generalize the function to make it flexible for new cases.

The details of the generated function depend on many factors, for example whether the state input and action output are continuous or if they are discrete, whether the mapping function is approximated while executing or previous to it, or whether the entire data set from the learning process will be kept.

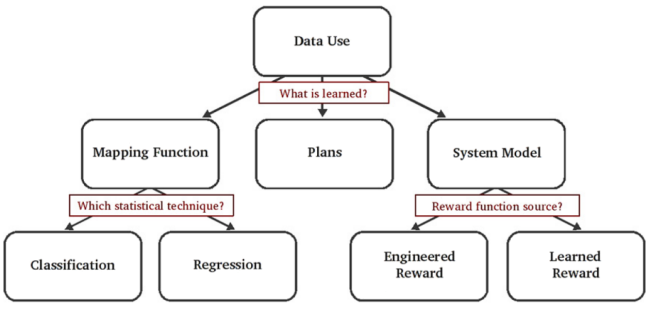

Fig. 6. The different categories of policy representation (taken from [5])

These details of mapping functions can be divided into two categories again, the category of *classification* for techniques producing discrete output and the *regression* category for techniques producing continuous output.

2.2.1.1 Classification

In classification, similar input is being grouped into discrete classes. Here, the input are the different robot states and the output are the robot's actions.
The actions can be at three different control levels: *basic motion control*, *action primitives* and *complex behaviours*. In [6], actually similar control levels are proposed, but they are only parted into two categories (there referred to as skill representation level). One is the low-level- or *trajectory* level representation, which corresponds to the basic motion control. The other one is the high-level- or *symbolic* representation, where "pre-defined motion elements"[6] are employed, which is similar to the action primitives and complex behaviours from [5].

Argall et al. [5] mention that the low-level-robot actions (basic motion control) include basic commands like forward or turning movements. They furthermore state that approaches which work on this level work for example with *Gaussian Mixture Models*, *decision trees*, *Bayesian networks* and *k-Nearest Neighbors (kNN)* classifiers.

If action primitives are used, they are often combined or sequenced together. Approaches on this level for example use *k-Nearest Neighbors* as well, but combine it with *Hidden Markov Models (HMMs)*.
Hidden Markov Models are also mentioned by [6] for the symbolic level and the more recent work of Zhu et al. [15] describes HMMs as a popular methodology for encoding and generalizing demonstrations. They state HMMs are a robust probabilistic method for encapsulating human movements, usually learning offline, but with online approaches available as well. They further explain that the HMM is a statistical model describing a Markov process with unobserved states, with resolving the hidden parameters for further analysing being the key problem in HMM. Trajectories demonstrated by human teachers can be can be represented as HMM with K states encoded by Gaussian mixture regression.

For the high-level behaviours, similar approaches can be used. [5] mentions using eigenvector representations for recognising gesture behaviours, HMMs, using the Bayesian likelihood method for selecting actions and the employment of Support Vector Machines for classifying behaviours.

2.2.1.2 Regession

The Regression technique maps demonstration states to continuous action spaces. Input (robot states) and output (actions) are similar to the classification technique, just that the output is continuous here.

Argall et al. [5] explain that, because continuous-valued output often results from combinations of several demonstrated actions, regression approaches typically employ low-level instead of high-level behaviours. The more distinct

difference in approaches using regression lies in whether the approximation of the mapping function happens at runtime or prior to it.

Regression techniques are distributed between those two extremes. *Lazy Learning* lies on one of these extremes, as here, functions are developed only when a current observation point needs a mapping. Simple lazy learning can work with kNN, more complex approaches use *Locally Weighted Regression* for example. Techniques which convert the recorded data to a new form before runtime and then use the converted data at runtime (e.g. for lazy learning techniques) are located more in the middle between the extremes. They can work with combinations of Gaussian and local linear model representations, for example, which can make approaches able to incrementally update the number of representative Gaussian and regression parameters while online. The extreme on the other side includes approaches which approximate the mapping function before starting any execution, making them independent of underlying data at runtime. But on a down-side, these techniques are more computationally expensive than the others prior to runtime. Neural Network techniques are used in these approaches, as well as statistical techniques.

2.2.2 System Models

Instead of focusing on approximating the underlying function, the system model approach focuses on finding a state transition model of the world and develops policies from that. For this approach, it can for example be helpful to use data which was collected about the environment during the demonstrations as this offers additional information about the world the robot is in.
Argall et al. explain that these types of approaches are typically used within the structure of Reinforcement Learning. From the data from the teacher's demonstration and the robot's potential additional exploration, a transition function is determined. A reward function is learned from the demonstrations or defined by the user and is, as mentioned above, used to develop policies from the transition model. In reinforcement learning, the goal is to maximize the cumulative reward over time, with the reward being represented by a value which can be calculated using the reward function. States and action selection and therefore also task execution are all done with respect for the reward. Reinforcement learning techniques usually do not generalize the states, therefore teacher demonstrations are needed for each state. Reward functions can be acquired using different techniques. Here, a distinction is made between *engineered* reward functions and reward functions *learned* from the teacher demonstration.

2.2.2.1 Engineered Reward Functions

Argall et al. state that reward functions are engineered in most PbD approaches. Engineering a reward function in this context means manually defining it.
As the reward functions are often build in a way where only few states give a reward value, while the others do not return any reward, it can be useful to highlight the actions which actually are rewarded to prevent the robot from too much

unsuccessful trial and error, thereby improving the performance of the learner. To highlight these areas of interest, the robot can for example be teleoperated to experience the rewarded states. There are also approaches which penalize negative behaviour in addition to reinforcing and demonstrating positive behaviour. Algorithms can also interact with other autonomous agents and exchange and incorporate advice from them to enhance and speed up the learning process, letting the algorithms learn cooperatively. This has however mostly been mostly studied in simulations. Real applications also have to consider the cost of demonstrating each action for each state. There is an approach which uses simulation to create an initial world model and, on the opposite, approaches which derive a model from demonstration, but then use a model simulation.

2.2.2.2 Learned Reward Functions

As defining an effective reward function by hand can be quite difficult, therefore the approach of learned reward functions focuses on learning a new reward function instead of manually defining it.

To do this, it is possible, for example, to associate great rewards with states which are similar to the demonstrated ones. On the other hand, it is also possible to let a human user individually reward tasks and let them recommend actions. Other rewardable goals are trajectories similar to the ones demonstrated by the teacher and positions which are close to the target. Using margin-maximization techniques to create a terrain cost map for finding paths as state-sequences is also presented as an approach. To resolve ambiguities, for example in reward functions, the principle of maximum entropy can be used. In contrast to the approaches mentioned until now, it is also possible to have a transition function additionally to the reward function.

2.2.3 Plans

In planning, instead of mapping states directly to actions, policies are represented as "a sequence of actions that lead from the initial state to the final goal state" [5].
Here, actions are often defined using a *pre-condition* to describe what has to happen before the action can be executed, and a *post-condition* to describe the result of the action. Often, additional information is given by the teacher in form of *annotations* or *intentions*.

Different algorithms which use planning differ in the way the connections between pre- and post-conditions are learned and in the question, if the teacher provides additional information. Argall et al. [5] mention approaches which use spoken dialog to demonstrate plans as well as verifying parts of plans, and other approaches which require teacher annotations, such as the overall intention of a task, but also feedback for drawing attention to particular domain elements.
Goal annotations are mentioned as well, being responsible indicating the demonstration goal and the points in time at which decisions about achieving this goal have to be made, as for well as providing binary feedback.

3 Methodology

In this section, the methods used for reviewing and categorizing the literature will be presented and an overview will be given about the attributes on which the approaches will be compared.

3.1 Literature Selection

The literature reviewed was selected based on several characteristics: firstly, the top search results for robot programming by design were included, as they should be highly significant. Secondly, a search for more recent works was conducted, as many of the previous found work was more than ten years old. To ensure a sufficient selection of works, the period for this search step was set to 2016 and following. Special attention was given to works which were cited often, as this implies their significance. Generally, some work was not considered for the detailed literature review, partly because it was outdated at more than twenty years old, or because it was focused in areas that are too specified for the general and widespread approach on the topic of robot PbD which this paper aims for.

As the VARobot project focuses on developing a virtual and augmented reality assisted robot programming environment, approaches which include the concepts of VR or AR were preferred to others. Furthermore, extra searches for these topics were conducted to make sure to include a sufficient selection of work on these topics.

It has to be remarked that survey works were looked into especially to gain a fundamental and broad understanding of robot programming by demonstration and that these surveys were used as base for the Foundations section and parts of the introduction. The surveys will however not be compared to actual approaches on single scenarios in Section 4, as their purpose is to give an overview over different approaches, not to present a specific one.

3.2 Comparison Criteria

As there is not much information about the technical nature of the VARobot project available at this point in time, it is difficult to decide which approaches are the most relevant. As virtual and augmented reality in combination with robot learning by demonstration was mentioned, the inclusion of each of these topics (VR and AR) will be part of the evaluation criteria. The robot operating system ROS was also talked about, therefore it will be considered as well.

Furthermore, the localisation of the approaches in the concepts and categories described in Section 2 will be included as well (as far as possible). The approaches will also be compared on the amount and type of equipment they need and on whether they are practically implemented or only a theoretical concept.

Additionally, it will be evaluated how many demonstrations each approach needs to learn from it, as this can be an important factor in the usability and user-friendliness of the approach. This is the case because the repetition of demonstrations, especially when necessary more than a few times, is very time-consuming as well as a tedious task for the user.

4 Literature Review

In this section, the approaches selected as described in Section 3 will be presented. After presenting them, the approaches will be compared on the criteria defined in Section 3.

4.1 Selected Approaches

In this section, seven approaches will be described. They will be ordered into the categories defined in Section 2 and it will be evaluated how far they fulfill the criteria defined in Section 3.

4.1.1 Robot Programming by Demonstration with Interactive Action Visualizations

The work by Alexandrova et al. proposes a PbD framework written in ROS which enables the user to teach a robot using kinesthetic teaching and voice commands [4]. As they are focusing on making robot PbD more user friendly and simple, their approach can learn a task from just one demonstration.
The framework saves the learned actions with respect to the robot's six degrees of freedom, the frame of reference (which consists of the relevant landmarks), and the robot's gripper state (open or closed). It also includes a GUI (see Figure 7), in which the user can see the learned actions (in form of the robot's poses and an indication to the landmark it is relative to), the landmarks and the order of the actions. There, the user can perform changes and improvements on the learned actions. They can delete landmarks and robot poses and change the reference frame and transformation of each pose. The approach was evaluated in a broad range of scenarios of object manipulation and was quite successful when validated with end-users, as all users managed to accomplish the tasks posed to them.

As mentioned above, the approach uses ROS and only needs one demonstration. It does however not mention AR or VR. In terms of the categories from Section 2, Alexandrova et al. use teleoperation through kinesthetic control and use plans for their policies. In terms of equipment they do not need much, only a kinect sensor mounted to the robot and a microphone headset for the teacher.

4.1.2 A ROS-integrated architecture to learn manipulation tasks from a single demonstration

Peppoloni et al. propose an approach for learning manipulation tasks in a single demonstration, which is integrated in ROS [12], similar to Alexandrova et al., but use a very different approach.
They define a set of primitives, which in this approach are locating a zone of interest, move between zones of interest, observing the environment state, and interacting with the environment (by manipulating objects). These predefined elements are mapped from the human demonstrator to robot skills. The robot

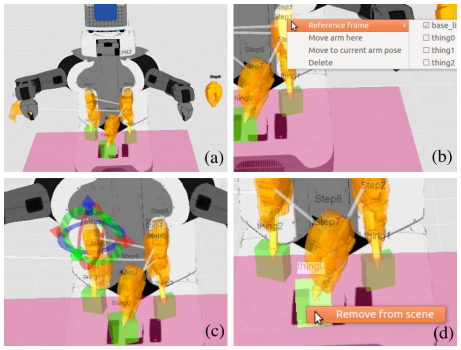

Fig. 7. The GUI of [4] showing several actions and options to manipulate them

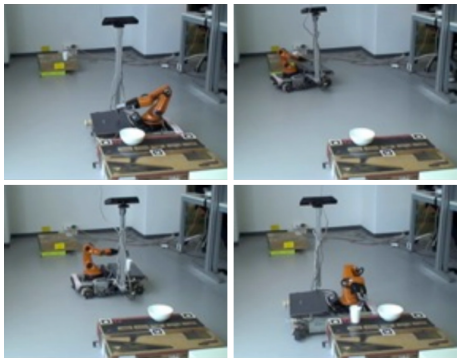

Fig. 8. The robot from [12] retrieving a cup from the palette in the back and placing it on the one in the front

learns new skills by watching a demonstrator do a task, while receiving vocal commands from them which indicate zones of interest locations and state changes. It saves this information in form of steps, where each step saves the current state of the environment (more specifically, the position of all objects). The robot then extracts the actions connecting these states by comparing them. When the robot is then asked to reproduce the task (like in Figure 8), it compares the saved action steps and tries to apply them to the detected environment. If steps are missing or are unnecessary because the situation is not exactly the one in the demonstration, the sequence of steps is adjusted by comparing the actual and the desired environment state and deriving the additional actions from it, or removing actions, respectively.

As the previous approach, this work needs only one demonstration and uses ROS, but does not use AR or VR. In terms of categorization however, they use external observation, as the robot observes the environment state, and a mapping function. The equipment requirements are moderate with only a kinect sensor.

4.1.3 Robust trajectory learning and approximation for robot programming by demonstration

Aleotti et al. [1] developed an approach on robust trajectory learning and approximation. They use a virtual reality glove to let the user do one or multiple demonstrations (of picking and placing an object) in a virtual environment.
For every operation, they trace the picked object's x, y and z coordinates to create a trajectory for each demonstration. If the teacher has performed multiple demonstrations, they are grouped into different sets representing alternative solutions using clustering.
Aleotti et al. use HMMs for selecting trajectories and Non-Uniform Rational B-Splines (NURBS) for trajectory approximation (see Figure 9). When only a

single demonstration is given, they then filter out high-frequency trajectory components (e.g. originating from noise or arm tremor of the user) and approximate a fitting trajectory using NURBS to get a compact, high-level representation. If multiple demonstrations were made, their framework uses implicit information as well to find out more about the intended motion.
Furthermore, it identifies the qualitatively different trajectories of the user. The high-level representation gained here can be manipulated by the user in a graphical environment to make local changes possible. After this is done, the learned task is performed in the simulated environment and has to be validated by the user before it is made available for being executed in a real environment.

This approach by Aleotti et al. does use VR, but not AR or ROS. It can work with one single demonstration, but can make use of more than one demonstration as well. Here, external observation is used as well, because, even though the user is wearing a virtual reality tracking glove and a motion tracking device is used, the approach observes the position of the virtual grasped object. For policy derivation, a system model developed, which models especially the state transition of the grasped object.

4.1.4 Programming manipulation tasks by demonstration in visiuo-haptic augmented reality

A newer work by Aleotti et al. from 2014 focuses on the inclusion of *Augmented Reality* (AR) in robot programming by demonstration [3]. They present an approach which uses visiuo-haptic AR to remotely teach a robot how to perform object manipulation tasks.
A lot of focus is placed on the AR part in this work, as several tasks have to be accomplished to make AR usable in robot PbD. For example, the different reference frames (of the robot, the workspace, the camera filming it and the user's haptic device) have to be coordinated (the different reference frames are illustrated in Figure 10). Furthermore, the recognition of objects in the workspace is quite complex. Aleotti et al. use a laser scanner to create scan the environment, generating point clouds. The points are then clustered using cluster extraction through a flood fill algorithm after executing outlier and plane removal. The point cloud clusters are then used for object recognition by comparing them to 3D models of the objects that could be in the workspace and selecting the model which minimizes the sum of squared distances between the points of the model and the clustered points.
The detected objects are then included into the AR environment either as static virtual bodies, which can not be moved, or as dynamic virtual objects. Dynamic virtual objects can be selected and deselected in the demonstration and the user can transform and rotate them while selected using a haptic device. This haptic device will also alert the user via force feedback if they try to pass through static objects.
After the first demonstration, the robot initialises a first task precedence graph, which contains a sequence of elementary actions and arcs, which define precedence relations between the actions. The actions are for example moving an

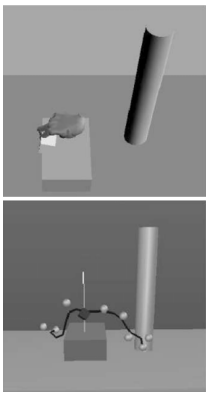

Fig. 9. The trajectory curve demonstrated in the first picture is approximated using NURBS, resulting in the curve in the second picture

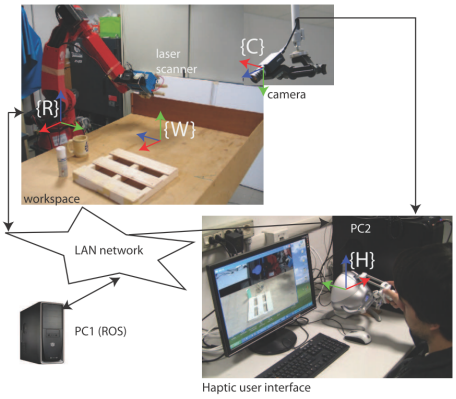

Fig. 10. The different reference frames in [3]: R for the robot, W for the workspace, C for the camera filming it and H for the user's haptic device

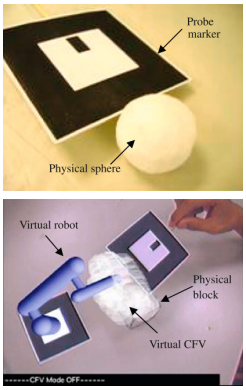

Fig. 11. A 3D marker with a sphere attached is used for defining the Collision-Free Volume in [11], the process is shown in the second picture

object next to or inside another one. With every new demonstration, the task precedence graph is updated to refine the set of valid precedence relations.

This approach does use AR and ROS, but not VR, and can work with one or multiple demonstrations. It makes use of a laser scanner and an affordable 3 DOF haptic device. Its correspondence category is external observation and uses a mapping function as a policy.

4.1.5 A novel AR-based robot programming and path planning methodology

Ong et al. propose an approach incorporating augmented reality as well, but they focus more strongly on the path planning methodology [11]. The approach relies fully on marker-based tracking and does not require a laser scanner and let the user experience AR using a head-mounted display.
Just as [3], different coordinate systems work together for the approach to work. Here, those belong to the camera, the robot's base, and its wrist (which is simulated by the user holding a marker, as a real robot is not involved here). In addition to the robot's wrist (or, more precisely, the marker the teacher is holding), there is a marker attached to the robot's base as well, both of which enable the approach to relate the coordinate systems.
As Ong's objective is to find methodology to generate a collision-free path for a robot moving an end-effector for non-contact tasks (e.g. arc welding or laser

cutting) along a visible path at a specific rotation, several steps must be taken. Firstly, a number of demonstrations of the desired curve has to be completed. Data points are collected from each curve at runtime, being shown to the user as visual feedback. The points can either be collected one at a time, with the user being able to reject the latest data point if it does not seem fitting, or they can be used in tracing mode, where the user can select between whole demonstrations, discarding bad ones all at once.
After this, the curves have to be learned. For this, the data can be parameterized either using simple parameterization using time or using Piecewise Linear Parameterization (PLP). The curve learning method then combines Bayesian neural networks and reparameterization to perform fitting according to the orthogonal distances between the data points and the model that they are fitted to. Ong et al. also propose to use a single hidden unit layer with Levenberg–Marquardt implementation and update the parameters after each iteration of the neural network.
Once the curve learning is done, a Collision-Free Volume (CFV) has to be found. For this, the user moves a marker with a sphere attached to it around the space where the robot has to move through. In this step, a virtual robot is shown so the user can assess if the robot would hit any objects and can reach all sections. After this, the user has to set the orientation of the end-effector as well to make it possible to simulate the whole path of the robot holding the end-effector and check if any collisions occur. In case there are collisions, the user has to change the end-effector orientation. The approach does produce an output which can be used for real robots, but it was not tested with one in this work.

The work by Ong et al. obviously supports AR, but does not mention VR or ROS. It also needs multiple demonstrations, but it works with only a HMD (which must have a camera though). In this approach, the external observation is used again, here strongly supported by the use of markers, and the policies are in form of a mapping function again.

4.1.6 Gesture based robot programming using ROS platform

Forgó et al. propose an approach which is mainly focused on introducing gesture based PbD using the ROS platform [8]. They developed a system for ROS detecting gestures using a LeapMotion sensor.
The framework was integrated with the ABB RobotStudio software, so the virtual environment in the software could be used. Furthermore, ABB RobotStudio is used to process the data which the ROS framework detected and applying it to its virtual robot representation. The approach can monitor the 3D palm position, the palm normal vector, the direction vector, and the 3D palm pose (pitch, yaw and roll) for a single hand, as well as all finger joints' positions. Once the sensor is calibrated with the virtual environment, these 3D parameters can be transferred on the (virtual) robot as needed. Additionally, the robot's gripper state can be directed based on the teacher's finger joint positions.
The proposed ROS framework then converts the data to be usable for the ABB RobotStudio. For demonstrating a task, the user now just has to move the vir-

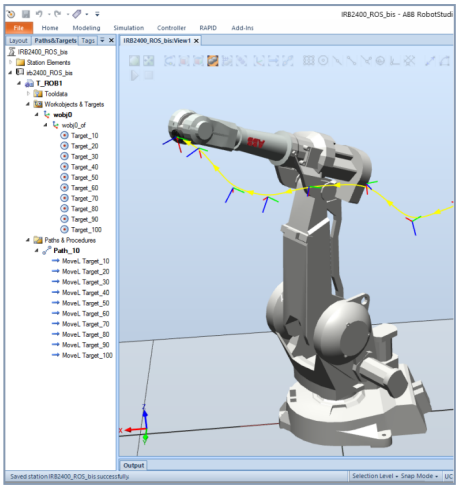

Fig. 12. A curve recorded by saving different positions indicates by the teacher's gestures, displayed in the ABB RobotStudio software

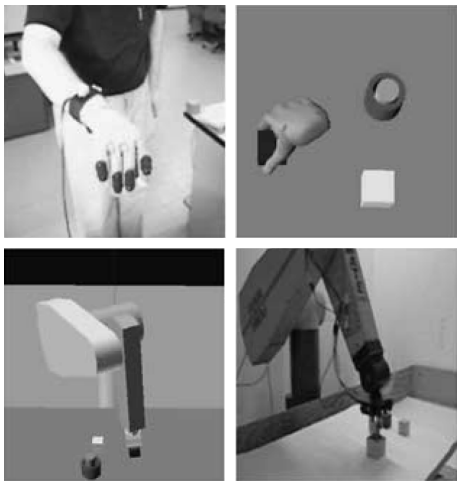

Fig. 13. From left to right: the dataglove used for transmitting data into the virtual environment, the representation in the virtual environment, a virtual simulation of the recorded task, execution of a learned task in the real workspace

tual robot in ABB RobotStudio using gestures. The user has to move the robot to the specific positions they want it in and use ABB RobotStudio's "Teach Position" feature to save the position. From these positions, a path is generated and stored and can be used for the robot again (the curve can be seen in Figure 12).

This approach does use VR and ROS, but not AR. It only needs one demonstration and only needs a hand-tracking sensor as hardware equipment. It is however built using the ABB RobotStudio software, so this software has to be used, which might reduce the choice of robots. In terms of categories, this approach actually employs shadowing, as the (virtual) robot follows the teacher's hand motions and saves its own states and movements using a mapping function.

4.1.7 Leveraging on a virtual environment for robot programming by demonstration

Another work by Aleotti et al. from 2004 focuses on the use of a virtual environment for robot PbD [2].
In their approach, the user uses a dataglove (see first picture of Figure 13) with a 3D tracker as an input device and the operators gestures are mapped directly to a 3D model of the hand in a virtual simulation of the real workplace (second picture of Figure 13). There also is a collision detection algorithm, which generates collision information between the hand and objects, making it possible to

provide feedback on object proximity to the user using vibration.
Using the data glove, the user executes the demonstration which is analyzed by the framework's task planner component. This component then segments the demonstration into a sequence of high-level primitives, namely into pick-and-place-, stacking- and peg-in-hole-operations. The segmentation is done whenever the grasping state changes to release an object.
The collected high-level tasks are then split into basic tasks, which include straight-line movements of the end effector, such as translations, and rotations along the z-axis. There are basic tasks for picking and releasing objects as well. Once the recognition is completed, the generated task is simulated to the user, still in the virtual environment (third picture of Figure 13). The user can watch to check whether the generated task is correct. If the user is not satisfied, they can delete the task. When a task is accepted, it is executed in the real workplace (see fourth picture of Figure 13).
Aleotti et al. explain that their approach makes it possible to add virtual fixtures to the virtual environment to make the programming by demonstration easier. This is accomplished in two ways in their approach: firstly, thresholds can be introduced to the tasks to create an acceptability zone. For example, this could prevent users from placing a virtual object below the ground plane (as this is not possible in a real environment). They state that with sufficient a priori knowledge, a virtual environment can guide the teacher towards semantically significant actions. Secondly, the vibration feedback can be used as a virtual fixture. The vibration can be different in strength and duration, making it possible to use is as indicator for different situations.

This work by Aleotti et al. works in VR. It does not employ AR or ROS (which did not exist yet when the approach was published [13]), but can learn in just one demonstration. It uses a dataglove to achieve sensors on teacher correspondence and plans for policy derivation.

4.2 Comparison of the Selected Approaches

The table in Figure 14 summarizes which criteria the presented approaches fulfill and how they fit into the categories presented in Section 2. It becomes visible that the approaches differ greatly, even when they are from the same group of authors (like [1], [3] and [2]).
Neither AR, VR or ROS are so popular in robot PbD that they are used in nearly all approaches, even though ROS seems to be used relatively often in newer works (it was only published in 2007 [13], so the older works could not have used it). Looking at the number of demonstrations needed to learn a task, it becomes visible that even recently, there are still approaches which need (or at least can use) more than one demonstration, but generally, works tend towards trying to have to use only one demonstration.

Regarding the categories from Section 2, it becomes visible that a relatively high amount of approaches use *Imitation* techniques (especially external observation), but we see that all techniques are represented at least once. The policy

Legend:	+	does include	AR	Augmented Reality
	-	does not include	VR	Virtual Reality
	m	multiple	ROS	Robot Operating System

Approach	AR	VR	ROS	Number of Demonstrations	Correspondence	Policy Derivation	Equipment	Year
[4]	-	-	+	1	Teleoperation	Plans	Kinect sensor Microphone Headset	2014
[12]	-	-	+	1	External Observation	Mapping Function	Kinect Cameras	2014
[1]	-	+	-	1 - m	External Observation	System Model	Virtual Reality Glove Motion Tracking Device	2006
[3]	+	-	+	1 - m	External Observation	Mapping Function	Laser scanner 3 DOF Haptic Device	2014
[11]	+	-	-	m	External Observation	Mapping Function	HMD (with camera)	2010
[8]	-	+	+	1	Shadowing	Mapping Function	Hand-Tracking Sensor ABB RobotStudio	2018
[2]	-	+	-	1	Sensors on Teacher	Plans	Dataglove	2004

Fig. 14. A summary of the presented approaches

derivation leans towards mapping functions, but also includes all categories. Looking at the necessary equipment, we see a very high variance, even though some similarities can be found. For example, the hand tracking sensor used in [8] fulfills a similar task as the data glove from [2] or the virtual reality glove from [1] due to the technological advancements.

Overall, it becomes visible that robot PbD can be done successfully in many different ways, allowing for great variance in techniques as well as equipment. Furthermore, there is still room for improvement and specialisation in many areas, for example in the area of AR inclusion in the learning process.

5 Conclusion

In this section, a short summary of this work will be presented in Section 5.1, then the results will be explained in Section 5.2 and an outlook on future work will be given in Section 5.3.

5.1 Summary

In this work, the topic of robot programming by demonstration was discussed. Its foundations were elaborated on and the categories of robot PbD were intro-

duced. The different combinations of record and embodiment mapping, namely teleoperation, shadowing, sensors on teacher and external observation, were described. Furthermore, the policy derivation techniques mapping function, system model and plans were elaborated on. After this, criteria for comparing different approaches in PbD were defined and a literature survey was executed, presenting different approaches and comparing the results.

5.2 Results

It became clear that robot PbD is a very wide field with many different possibilities and specialisations. Furthermore, new options open up through technical progress, which makes new approaches possible, that may not have been much looked into before.

Especially in relation to the VARobot project, AR and VR seem to offer great potential which was not exploited yet, in particular if newer technologies are taken into account. Several approaches discussed here have shown the possibilities of including AR and VR into robot PbD. In addition to this, it became apparent that there is already some work on using ROS in robot PbD, which could be used as an inspiration for the VARobot project.

5.3 Future Work

To properly use robot programming by demonstration in the context of VARobot, firstly, the current state and available tools have to be assessed. Furthermore, a more detailed plan for the project has to be made. Thereafter, a decision based on the information presented here can be made, determining which category of approach is the most suitable and how to exactly include AR and VR to achieve the highest possible usability and efficiency.
To make an informed decision, it will also be necessary to gain more knowledge about the topics which play into robot programming by demonstration, such as robot programming as well as VR/AR topics as they are supposed to be included.

References

1. Aleotti, J., Caselli, S.: Robust trajectory learning and approximation for robot programming by demonstration. Robotics and Autonomous Systems **54**(5), 409–413 (2006). https://doi.org/10.1016/j.robot.2006.01.003
2. Aleotti, J., Caselli, S., Reggiani, M.: Leveraging on a virtual environment for robot programming by demonstration. Robotics and Autonomous Systems **47**(2), 153 – 161 (2004). https://doi.org/https://doi.org/10.1016/j.robot.2004.03.009, http://www.sciencedirect.com/science/article/pii/S0921889004000454, robot Learning from Demonstration
3. Aleotti, J., Micconi, G., Caselli, S.: Programming manipulation tasks by demonstration in visuo-haptic augmented reality. In: 2014 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE) Proceedings. pp. 13–18 (Oct 2014). https://doi.org/10.1109/HAVE.2014.6954324
4. Alexandrova, S., Cakmak, M., Hsiao, K., Takayama, L.: Robot programming by demonstration with interactive action visualizations (07 2014). https://doi.org/10.15607/RSS.2014.X.048
5. Argall, B., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems **57**, 469–483 (05 2009). https://doi.org/10.1016/j.robot.2008.10.024
6. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot Programming by Demonstration, pp. 1371–1394. Springer (01 2008)
7. Duan, Y., Andrychowicz, M., Stadie, B.C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., Zaremba, W.: One-shot imitation learning (2017)
8. Forgo, Z., Hypki, A., Kuhlenkoetter, B.: Gesture based robot programming using ros platform. In: ISR 2018; 50th International Symposium on Robotics. pp. 1–7 (June 2018)
9. Kreuziger, J., Kaiser, M., Dillmann, R.: Robot programming by demonstration (rpd) - using machine learning and user interaction methods for the development of easy and comfortable robot programming systems. 25th International Symposium on Industrial Robots (ISIR '94) (08 1994)
10. Levas, A., Selfridge, M.: A user-friendly high-level robot teaching system. In: Proceedings. 1984 IEEE International Conference on Robotics and Automation. vol. 1, pp. 413–416 (1984)
11. Ong, S., Chong, J., Nee, A.: A novel ar-based robot programming and path planning methodology. Robotics and Computer-Integrated Manufacturing **26**(3), 240 – 249 (2010). https://doi.org/https://doi.org/10.1016/j.rcim.2009.11.003, http://www.sciencedirect.com/science/article/pii/S0736584509001100, product Design and Manufacturing Systems 07 on Advanced Robotics and Machine Design
12. Peppoloni, L., Di Fava, A., Ruffaldi, E., Avizzano, C.: A ros-integrated architecture to learn manipulation tasks from a single demonstration. vol. 2014 (08 2014). https://doi.org/10.1109/ROMAN.2014.6926308
13. ROS: History. https://www.ros.org/history/, accessed: 05.06.2020
14. Tykal, M., Montebelli, A., Kyrki, V.: Incrementally assisted kinesthetic teaching for programming by demonstration. In: 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI). pp. 205–212 (2016)
15. Zhu, Z., Hu, H.: Robot learning from demonstration in robotic assembly: A survey. Robotics **7** (04 2018)