One-shot Robot Programming by Demonstration by Adapting Motion Segments

Christian Groth and Dominik Henrich†
Lehrstuhl für Angewandte Informatik III
Universität Bayreuth, D-95445 Bayreuth, Germany

***Abstract*— In order to provide robots for a wide range of users, they must be easily programmable. This is addressed by various approaches of programming a robot by demonstration, where the user guides a robot through the task.**
We propose an approach that adapts a once demonstrated trajectory to a new situation. The system extracts the relevant reference frames using a weighting function. Then, the demonstrated trajectory is segmented with respect to these frames and different frame-dependent strategies for reproduction are applied. The approach is capable of adapting simple point-to-point paths, more complex tasks like pick-and-place operations, and manipulations where objects are used as tools. Additionally, the system extracts the required resources for a task to integrate the approach into a behavior-based system, which manages new programmed behaviors and selects a suitable behavior in a new situations. The approach is computationally efficient and comprehensible for the user. We evaluated the approach qualitatively and quantitatively using cross validation.

I. Introduction and State of the Art

Since traditional robot programming is tedious and only feasible by experts, there is a strong need for concepts to easily program a robot. Programming by Demonstration (PbD) is such an approach, which allows even non-experts to program a robot just by guiding it through the task. The system extracts relevant task features, allowing the reproduction of the task in new situations.
The demonstration can be provided on a symbolic level or on a trajectory level. On the symbolic level, often predefined primitives are used, such as (dynamic) movement primitives or skills, which act as building blocks for complex tasks. The challenge is to identify these primitives, find appropriate parameters, and compose them [1] [2] [3]. This has been investigated for just one demonstration in [4] [5] and [6]. If multiple demonstrations are available, the recognition of primitives may be improved and even alternative task executions can be provided [7] [8]
On the trajectory level, usually multiple demonstrations are available. This allows the application of various machine learning algorithms. A good survey is provided by [9]. More recent research can be found in [10] [11] [12] [13] [14]. Only little research addresses the problem of adapting the trajectory of a single demonstration to a new situation. But, existing approaches rely either on additional information like speech input [15] [16] or on manual post-processing of the trajectory [17]. A very promising approach in [18] uses a computationally expensive global warping technique to adapt a path to a new scene, but it is not able to adapt the robots orientation.
Since predefined primitives may limit the space of motions that can be programmed, we concentrate on the trajectory level. In order to reduce the teaching burden for the user, we investigate how we can program useful tasks to the robot, using just a single demonstration. In contrast to existing work, we give an analytic solution to trajectory adaption, that utilizes local features. I.e. not all of the features from the demonstration must be present to allow the reproduction to begin. Instead, the system adapts the demonstration piecewise to the new situation.
Additionally, we want to integrate the approach in a behavior-based system. Contrasting, in most approaches the execution is triggered by the user. Exceptions can be found in the mobile robotics domain [19] [20]. Since a useful robot in the household should not need to learn the task over and over again before execution [21], it has to store the behaviors and select a suitable behavior for each situation by itself. Therefore we will integrate the approach into our behavior-based system [22].
Summarizing, many approaches to Programming by Demonstration that rely on multiple demonstrations exist. Approaches based on a single demonstration are computationally expensive, cannot adapt the robot orientation, require manual post-processing, or can only recognize predefined actions (primitives). Additionally, the execution is mostly triggered by the user. In this work, the main contribution is a Programming by Demonstration approach on trajectory level that gives an analytic solution to adapt a single demonstration to a new situation. The algorithm is able to cope with new robot configurations and new object positions and orientations. Furthermore, the systems stores previous demonstrations and is able to select a demonstration, that is suitable to be adapted to the new situation.
The remainder of this paper is organized in a quite standard manner: Problem Formulation, Approach, Experimental Results, and Conclusions.

II. Problem Formulation

The aim of the approach is to enable the robot to execute an adapted version of a task in a new situation under new constraints, like a changed starting pose and changed

†This work has partly been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant agreement He2696/15 INTROP.

object positions and orientations. The user provides just one demonstration for every task. The adapted trajectory should fulfill the task in the new situation and have similarity with the corresponding demonstration. Furthermore, the reproduced motion should be identical to a demonstration under unchanged constraints, since it is the ground truth from the user.

In this work, we regard robotic manipulators mounted e.g. on a table or a mobile platform with a certain range to operate. A demonstration consists of a 6-dimensional trajectory $T = \{t_0, \ldots, t_{n-1}\}$ with $n \in \mathbb{N}$ of the robot's tool frame t_i in 3D space and a set of 6-dimensional reference frames $F = \{f_0, \ldots, f_{m-1}\}$ with $m \in \mathbb{N}$, where each frame defines a position and rotation in 3D space. A sample (via-point) t_i can carry additional tool operations, like the gripper position. A single reference frame f_i is defined by, e.g., an object, a marker, or a part of the robot. We will denote reference frames defined by objects as object frames.

In order to adapt a demonstration to a new situation, we need a mapping of every sample $t_i \in T$ to the set of relevant frames $\tilde{F}_i$:

$$\forall t_i \; \exists \; \tilde{F}_i \subseteq F$$

This mapping tells us to which frames the current sample is related to. We realize this mapping via a $m \times n$ weight matrix W where each matrix element w_{ij} carries the weight of a frame f_i with respect to a sample t_j. Thus, W provides the relevance of each frame w.r.t. every sample.

Moreover, we need a function $rep()$ that calculates a robot trajectory T^r from a demonstrated trajectory T, the existing frames in demonstration F, and the corresponding frames F^{r} in the new situation, which may have changed positions and orientations. Of course this function will need W to calculate the new trajectory:

$$T^r = rep(T, F, F^{\mathrm{r}}, W)$$

Based on a demonstration, this function will generate a trajectory in the new situation while considering the weight of each frame to every sample. The calculation of W and the $rep()$ function are discussed in the next section.

III. Approach

A. Concept

As mentioned in Section II we need to calculate W in order to allow the function $rep()$ to reproduce the trajectory to the new situation. Of course there is no free lunch. If we provide less information, we need to make some assumptions to calculate W.

First assumption: Most of the useful robot activities involve objects. Objects can be manipulated or act as tools to manipulate other objects. Thus, here every object's initial position and orientation in the scene will define a reference frame. Another frame will be defined by the robot's tool of the initial pose.

Second assumption: Every sample point t_j of the demonstration can be associated with only a single reference frame f_i:

$$\forall_j : t_j \rightarrow f_i.$$

This will ease things a lot, because now we may divide the trajectory into segments with every segment being exclusively related to an object or to the robot itself.

With these assumptions, we can calculate the weight of a reference frame f_i with respect to a sample t_j by means of a distance function $dist(f_i, t_j) \geq 0$. The distance function may be based on spatial features like the Euclidean distance between t_j and f_i, on temporal features, on a global homogeneity criteria or even on social cues.

We consider reference frames with smaller distance to be more important. Therefore we define a $m \times n$ reference matrix R with elements r_{ij} by:

$$r_{ij} = \frac{1}{1 + dist(f_i, t_j)}$$

Exploiting the second assumption, we use R and define a $m \times n$ segmentation matrix U with elements u_{ij} by

$$u_{ij} = \begin{cases} 1, & \text{if } r_{ij} = \min\limits_k(r_{kj}) \text{ and } r_{ij} < d_{\max} \\ 0, & \text{otherwise} \end{cases}$$

We assume $r_{ij} \neq r_{kj}$ for $i \neq k$. For the unlikely case of an identical distance of a sample to different frames, we select by random. The threshold $d_{\max}$ discards the influence of a too distant frame f_i. This means, the reference frame with the smallest distance within a threshold $d_{\max}$ is considered relevant for the sample and all others are considered irrelevant. The computation of $d_{\max}$ will be discussed in Section IV-A. In U, some of the columns will only consist of zeros. We have to decide which reference frames are relevant for these. We assume that these trajectory samples belong to transfer motions between the preceding and the subsequent reference frame. Therefore, we have to assign these samples to the preceding and subsequent frame by adjusting the matrix elements which refer to these frames. Based on U, we can finally define W. We define for every c the column w_c of W by

$$w_c = \begin{cases} u_c, & \text{if } \sum_{i=0}^{m-1} u_{ic} = 1 \\ 0.5 \cdot u_p + 0.5 \cdot u_s, & \text{otherwise} \end{cases}$$

Here, u_p denotes the nearest preceding column and u_s the nearest subsequent column. These columns are defined by

$$\sum_{i=0}^{m-1} u_{ip} = 1, \quad |p - c| = \min, \quad p < c$$

and

$$\sum_{i=0}^{m-1} u_{is} = 1, \quad |s - c| = \min, \quad s > c$$

The columns u_p and u_s determine the relevant frames for the transfer. For better understanding, we give a short example

of W as

$$W = \begin{bmatrix} 1 & 0.5 & 0.5 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0.5 & 0.5 & 1 & 0.5 & 0.5 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & \end{bmatrix}$$

For the first sample t_0, the relevant reference frame is f_0. For sample t_3 it is f_2. Between these samples a transfer motion is assumed and a weighting of 0.5 for frames f_0 and f_2 is set. If there is no subsequent frame, the samples are assigned to the last referenced frame, since it could be some detach-movement from an object.

As we assume that we can assign every sample to one reference frame, we will introduce *transfer frames*, which are calculated from the subsequent and preceding reference frame if the mapping is ambiguous. The construction of these transfer frames will be shown in the next section.

Now, if we assign every sample to the highest weighted reference frame and the ambiguous samples to the transfer frames, we receive a segmented trajectory. Some of the segments are related to existing frames alternated with transfer segments. The segmentation may seem over-complicated at first glance. But there are some advantages against a conventional distance based segmentation. First, we do not have to care if the trajectory starts or ends in an object frame or in a transfer frame. We can just apply the calculation rules. Second, if the trajectory leaves an object frame at t_k and returns to it at t_m, the algorithm will also assign the samples between t_k and t_m to the object frame. This is e.g. useful if the robot executes handling on an object and has to depart from the object for the handling.

The adaption of the segments is described in the next sections.

B. Demonstration Phase

As described in the last section, we segment the demonstrated trajectory into subtrajectories, each associated to either an object frame or to a transfer frame.

All samples within the distance threshold are assigned to an object frame f_i. As stated, every frame holds at least a spatial coordinate system. We use the coordinate system inherent to the object, which is unambiguous in our case. The system should reproduce the trajectory segments associated to objects in a very exact way, since the object could be manipulated by this motion.

All samples that are not mapped to object frames, are mapped to transfer frames (Figure 1). We denote the set of all transfer frames as V. The reproduction of the transfer motion must satisfy the constraints of the new situation. Since a transfer motion depends on the preceding and subsequent frame, we calculate the transfer frame from the origins O_p and O_s of these frames. We also make use of the cutting points $p_{\text{cut},s}$ and $p_{\text{cut},p}$ of the object frames' hulls, defined by $d_{\max}$, and the connection line between O_p and O_s. The transfer frame is an orthogonal right hand coordinate system x_V, y_V, z_V with its origin at $p_{\text{cut},p}$ and calculated as

$$x_{\text{V}} = p_{\text{cut},s} - p_{\text{cut},p}$$
$$x_{\text{V}} \perp y_{\text{V}}, x_{\text{V}} \perp z_{\text{V}}, y_{\text{V}} \perp z_{\text{V}}$$
$$\|y_{\text{V}}\| = 1, \|z_{\text{V}}\| = 1$$
$$\angle(z_{\text{V}}, g) = \min$$

where g is the negative gravity vector pointing up.

After having calculated the object and transfer frames,

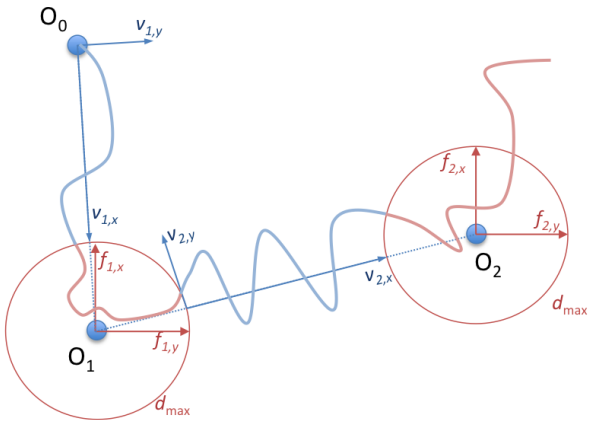


Fig. 1. 2D example of generated frames using a spatial distance function. The demonstration consists of two objects O_1 and O_2 and the first tool center point O_0. The blue parts of the trajectory are transfer motions, while the red parts relate to the objects. The associated frames are marked in the same colors.

we transform each sample t_j of the demonstrated trajectory T into its frame. First we separate the trajectory into the segments of points $\langle T_k \rangle_{\text{frame}}$ by $T = \langle T_0 \rangle_{\text{world}} \langle T_1 \rangle_{\text{world}} \dots \langle T_{n-1} \rangle_{\text{world}}$. I.e. all points remain relative to the world coordinate system. Each segment T_k represents the membership to the same frame, which is provided by the weight matrix W. Now we can transform the points into the object frames F and transfer frames V. We use the transformations ${}^{f_i}M_{\text{world}}$ for each frame $f_i \in F \cup V$ and denote the new transformed segments T_k as

$$\langle T_k \rangle_{f_i} = {}^{f_i}M_{\text{world}} \langle T_k \rangle_{\text{world}} .$$

C. Reproduction Phase

In reproduction phase we extract the reference frames F^{r} for all objects that correspond to the objects of the demonstration by means of a matching function, which is described in [22]. The transfer frames V^{r} are created analogous to the demonstration phase in Section III-B. Afterwards we extract the transformations ${}^{F^{\text{r}}}M_{\text{world}}$ for all frames F^{r} analogous. With this, we can reproduce the trajectory in the new situation by transforming the segments T_k from the frames into the world frame by

$$\langle T_k \rangle_{\text{world}} = {}^{f_i^{\text{r}}}M_{\text{world}}{}^{-1} \langle T_k \rangle_{f_i}$$

with $f_i^{\text{r}} \in F^{\text{r}} \cup V^{\text{r}}$.

When reproducing the trajectory in a new situation, we may encounter some harsh changes due to strongly rotated

or displaced objects. To generate smooth trajectories, we introduce a blending between the transfer frames and the object frames (see Figure 2). For this, we reuse the distance function from Section III-B with a higher threshold d_{blend}, which will enlarge the space. For all transfer samples which would be mapped to the object frame with threshold d_{blend} but are not mapped with threshold $d_{\max}$, we blend the trajectory according to the transfer frame and the trajectory in the object frame. Thus, for all t_j with $d_{\max} < dist(f_i, t_j) < d_{\text{blend}}$ we recalculate the reproduced sample point t_j^{r} as

$$t_j^{\text{r}} = blend(d_{norm}) \cdot {}^{v^{\text{r}}}M_{\text{world}}{}^{-1} \cdot {}^{v}M_{\text{world}} \cdot t_j + (1 - blend(d_{norm})) \cdot {}^{\text{world}}M_{f_i^{\text{r}}} \cdot {}^{f_i}M_{\text{world}} \cdot t_j$$

The v and v^{r} are the transfer frames in the demonstration and reproduction. The $f_i \in F$ and $f_{\text{i}}^{\text{r}} \in F^{\text{r}}$ are the object frames in the demonstration and in the reproduction. The first term is the position in the transfer frame. The second term is the original sample, treated as if it was related to the object frame.

The normalized distance d_{norm} is calculated from the sample's distance to the origin of the nearest object frame f_i as

$$d_{norm} = \frac{dist(t_j, F^{\text{r}}) - d_{\max}}{d_{\text{blend}} - d_{\max}}.$$

To get a smooth transit, we use the sigmoid function with empirically determined parameters

$$blend(d_{norm}) = \frac{1}{1 + e^{-16 \cdot (d_{norm} - 0.5)}}$$

with a relevant range of values between 0 and 1. When the trajectory is reproduced by the robot, we will need an interpolation of the trajectory due to eventually large distances from the scaled trajectory points. We use piece-

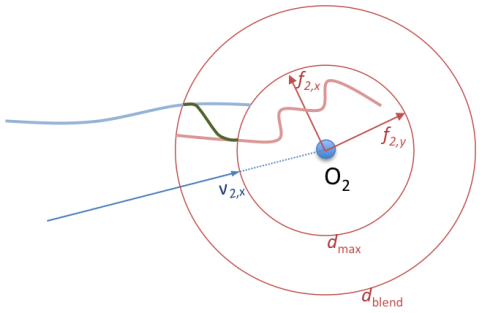


Fig. 2. 2D example of the blending (dark green) between an object frame (red) $f_{o,2}$ and a transfer frame (blue).

wise linear interpolation, due to a high number of available samples.

Summarizing, the adaption of a demonstrated trajectory is accomplished as follows: First, the demonstrated trajectory is divided into segments, based on the reference frames. Each segment is assigned to a reference frame or a transfer frame. In reproduction, the segments are transformed according to their displaced and rotated reference frames. This causes the segments, which belong to transfer frames not only to be rotated and displaced as well, but also to be stretched or shortened. The blending will take care of smooth transitions between the reproduced segments.

D. Multiple Behaviors

In [22] we already presented a behavior-based system, where concepts from modern operating systems domain have been adapted to the robotics domain. The system is able to execute state-machine-based behaviors by concurrent resource management. It is possible to interrupt and to resume behaviors safely. Here, we transform the sequence of subtrajectories into a behavior. Multiple of these behaviors can be triggered and executed in parallel by our system. In brief, a behavior is modeled as a Mealy machine $B = [S, C, A, \delta, \omega, s_0, s_F]$, where S denotes a set of states, s_0 and s_F denote the initial and final state, A denotes a set of actions and C a set of conditions. The transition function δ is given as $\delta : S \times C \to S$ and the output function ω is given as $\omega : S \times C \to A$. In this application every action $a \in A$ of a state stores the current subtrajectory in the corresponding frame ($a_k = \langle T_k \rangle_{\text{f}_k}$). As transition condition c, we take the objects from the demonstration which were used to construct the corresponding frame ($c_k = \{f_k, [f_{k+1}]\}$). If it is an object frame, we set the second condition $[f_{k+1}]$ to the empty condition ε. In execution, the system uses matching techniques, which are described in [22], to identify the corresponding objects needed for the transition condition. In this case it will return objects of the same type as in the demonstration. It will extract the frames and reproduce the trajectory according to the new frames. The system can switch between these behaviors by saving the current state and preemption of needed resources. It will also care for the blending between the frames.

The system is also capable of executing available tasks segment-wise. I.e. if we cannot reproduce the complete demonstration due to missing objects, the system will reproduce the task as far as possible, switch to another task, and resume to the prior task, when the rest of the demonstration can be reproduced. Through the segmentation of the trajectory and the explicit mapping to a frame, we can make full use of the approach in [22].

IV. Experimental Results

A. Experiments

We tested our approach in an experimental setup containing a robotic arm mounted on a table. This setup is favorable for applications in domains like laboratories, households, and workshops. All experiments are performed by a Kuka LWR IV Robot with 7 degrees of freedom equipped with a Schunk PG70 gripper. A top-mounted RGB-D camera is used to detect objects in the workspace. The trajectory is recorded while the user guides the robot through the task.

To evaluate the approach even with a simple distance function, we use the spatial Euclidean distance as criteria:

$$dist(t_j, f_i) = ||t_j - O_{f_i}||$$

where O_{f_i} is the origin of the reference frame f_i in 3D space. We set $d_{\max} = \min(2 \cdot s_{\max}, o_{\min})$, where $s_{\max}$ is the largest diameter of each of the objects bounding spheres and $o_{\min} = \min ||0.5 \cdot (O_{f_i} - O_{f_k})||$ for $\forall i, k, i \neq k$. We choose $2 \cdot s_{\max}$ because of the possible usage of an object as a tool (see the cooking experiment), but we have to avoid ambiguous assignments of samples to reference frames. The blending threshold is not crucial since it will only provide smooth transitions. An empirical estimated value of $d_{\text{blend}} = \min(1.5 \cdot d_{\max}, o_{\min})$ works fine in typical tabletop applications. Future work may also calculate d_{blend} from the magnitude of change in orientation and position compared to the demonstration.

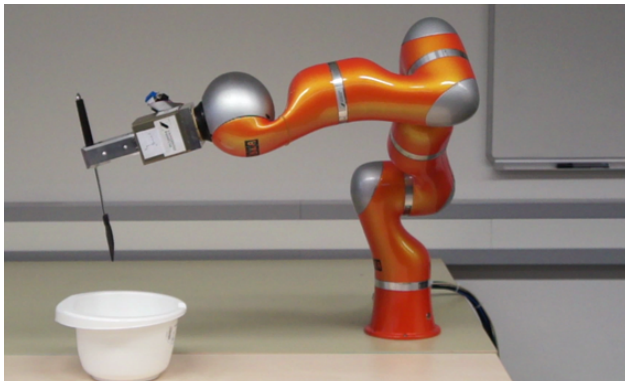


Fig. 3. Kuka LWR used in our experiments.

We performed two types of experiments: a quantitative cross validation against human demonstration of the approach (Experiment 1) and a qualitative evaluation (Experiment 2). A video with further applications is available on our website[1].

1) Experiment 1: The first experiment is meant to give a quantitative evaluation of the approach. Two different tasks were demonstrated by the user. The first task is a general grasping. The second one is a pick-and-place task. We demonstrated task T1 in four different ways (series S1 - S4) with 20 repetitions each and task T2 in three different ways (series S5 - S7) with 20 repetitions each. This results in $80 \cdot 80 + 60 \cdot 60 = 10.000$ cross validations. The series of T1 are inspired by [18] but modified due to the partially algorithmic-specific character. Task T1 consists of the following 4 series:

- Series S1 is a straight and direct grasping movement.
- Series S2 is based on S1, but the whole scenario is rotated by 90° around vertical up.
- Series S3 is based on S1, but the target is rotated by 90° around up.
- Series S4 is a grasping with arbitrarily chosen 3D orientations and positions of the robot and the target.

In every series, the constraints stay the same, but variations due to the different user demonstrations arise. In addition to T1, the series of task T2 shall demonstrate the applicability of the approach in more complex tasks.

[1] http://www.ai3.uni-bayreuth.de/projects/introp/

- Series S5 is a typical pick-and-place task where the robot returns to the initial position.
- Series S6 is based on S5, but the targets are relocated on the table and rotated arbitrary around up.
- Series S7 is based on S5, but with arbitrary chosen 3D positions and orientations of the targets and the robot.

We do a pair-wise comparison of each demonstration and every possible reproduction of both tasks. Therefore we take the reference frames F_{input} of a demonstration (input) and reproduce another demonstration to these new positions (output). This is achieved by setting the reference frames F_{input} from the input as reference frames for reproduction to the output $F^r_{output} = F_{input}$. Afterwards, we compare the reproduced output trajectory with the human demonstrated input trajectory and calculate two performance metrics. In analogy to [18] we calculate the mean squared difference MSD as

$$MSD = \frac{1}{N} \sum_{i=1}^{N} ||p_i^r - p_i||$$

where p_i^r is the position of the reproduced sample (output) and p_i is the input sample position. The calculated values will give insight on how close the reproduced trajectory is to a human demonstration.
We also calculate the correlation coefficient R^2 as

$$R^2 = \frac{\sum_{i=1}^{N}(p_i - \bar{p}) \cdot (p_i^r - \bar{p}^r)}{\sqrt{(\sum_{i=1}^{N}(p_i - \bar{p})^2) \cdot (\sum_{i=1}^{N}(p_i^r - \bar{p}^r)^2)}}$$

where $\bar{p}$ and $\bar{p}^r$ are the arithmetic means of p_i and p_i^r. The correlation coefficient will indicate how similar a reproduced path is to a human demonstration.
Both metrics require input vectors of the same length. Therefore we interpolate the shorter trajectory using a hermite spline.

2) Experiment 2: The second experiment is meant to evaluate the qualitative results of the algorithm. We provided three different demonstrations. First: A user gives a demonstration, where a pen is used to draw an arrow on a target (T3). The second demonstration shows a pick-and-place task, with a curling movement between the origin and destination (T4). The third demonstration shows the stirring of a cooking application (T5). A spoon is picked and used to stir the ingredients in a bowl. The spoon is placed back afterwards. The goal of the experiment is the reproduction of the tasks with changed orientations and positions of the targets while the characteristic movements are kept.

B. Results

Regarding Experiment 1, for the case of the 140 self-mappings, where a demonstration is adapted to identical reference frames, we consequently receive $MSD = 0$ and $R^2 = 1$. The system will act like in playback mode, which was one of our requirements from Section II since it reflects perfectly the users intended trajectory.
Tables I to IV show the arithmetic mean of the performance metrics for each group of input / output series.

TABLE I

MSD FOR ADAPTED SERIES OF TASK T1

	input			
output	S1	S2	S3	S4
S1	0.000238	0.00421	0.00369	0.00324
S2	0.00428	9.89e-05	0.00815	0.00895
S3	0.00405	0.00767	0.000648	0.00385
S4	0.00385	0.00564	0.0045	0.000638

TABLE II

R^2 FOR ADAPTED SERIES OF TASK T1

	input			
output	S1	S2	S3	S4
S1	0.996	0.935	0.909	0.904
S2	0.951	0.998	0.811	0.868
S3	0.851	0.923	0.981	0.883
S4	0.892	0.896	0.909	0.974

Regarding task T1, we notice a minimum correlation coefficient of $R^2 = 0.81$. This result gives an impression of how similar and close the adapted trajectories are to human demonstrated grasping. We received the best results for the adaption of series S1, but still get very good results for the adaption of the most complex case of series S4.

Regarding task T2, we receive lower correlation, but still 78% show results of $R^2 \geq 0.7$. The problem is the length of the task. The demonstrations differ much more from each other than in the simple grasping case. Therefore we instantly get lower R^2 and higher MSD. But nevertheless, the algorithm produces satisfying results and is able to adapt the execution of the task to a new situation even with changed starting position of the robot. The results show, that one-shot programming can be useful, when it comes to robot programming by demonstration.

We are not able to cross-validate demonstrations from task T1 to task T2 due to the nature of the algorithm. The algorithm expects the resources from the demonstration to be present in the reproduction, because it utilizes the resource inherent reference frames for adaption. Since more objects are involved in task T2 than in task T1, we cannot adapt the trajectory. This may be a point for future work.

TABLE III

MSD FOR ADAPTED SERIES OF TASK T2

	input		
output	S5	S6	S7
S5	0.00202	0.017	0.0277
S6	0.0168	0.00243	0.0102
S7	0.0267	0.0101	0.00172

Figure 4 and 5 show the qualitative results of the approach for demonstrations T3 and T4. Figure 4 shows the reproduction for displaced and rotated targets. The distance between the reference frames increases counter-clockwise for each

TABLE IV

R^2 FOR ADAPTED SERIES OF TASK T2

	input		
output	S5	S6	S7
S5	0.965	0.733	0.51
S6	0.789	0.954	0.784
S7	0.588	0.811	0.968

reproduction. One can still recognize the curling very well, what indicates a good preservation of the transfer motions. The orientation of the robot is also adapted accordingly to the corresponding frames. Although the relevant objects are displaced and rotated, the trajectory near them stays the same. This can be observed even better in Figure 5. After a bowed transfer to the target, an arrow is drawn on the target. Here, we can observe the reproduction of the desired motion near the target while keeping the characteristic bow of the transfer motion. The figures additionally show the adaption to different z-levels, which is also handled very well by the algorithm. In Figure 6 we can see the result for the adapted

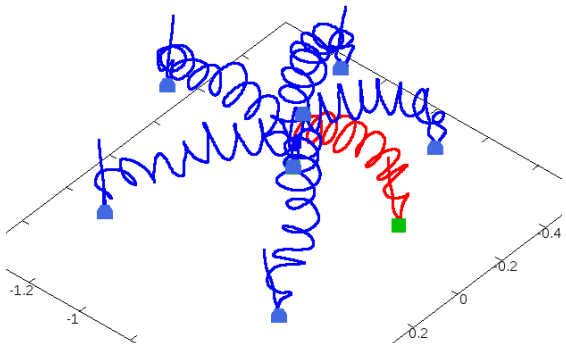
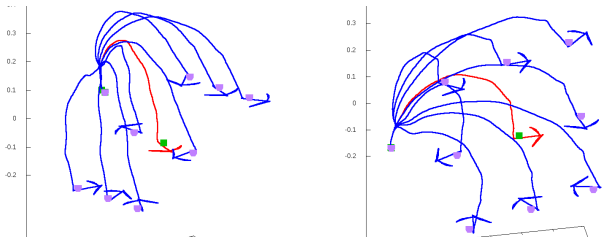


Fig. 4. Demonstration (trajectory red, objects green) and reproductions with changed positions and orientations (trajectory and objects blue) of T3

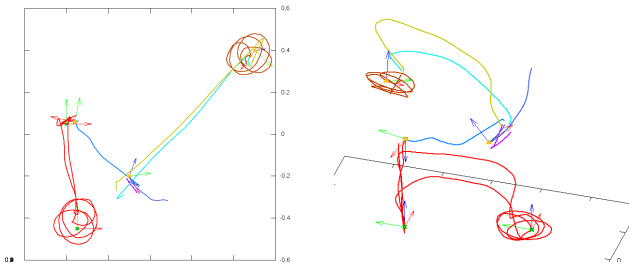


(a) (b)

Fig. 5. Demonstration (red) and reproductions with changed positions and orientations (blue) of T4

stirring task. The system is successfully able to pick up the spoon and stir inside the bowl. Additionally it is capable of reproducing the trajectory segment-wise. I.e. as long as only the spoon is present for reproduction, it will approach and pick up the spoon. As soon as the bowl is present, the transfer to the bowl is calculated and the execution is continued. This integrates very well into out behavior-based

approach [22]. The overall computation effort is small, thus,



(a) (b)

Fig. 6. Top view (a) and side view (b) of demonstration (red) and reproductions with changed positions and orientations (colored segments) of T5

the adapted trajectory is available instantly as soon as a new situation is presented to the programmed system. In future work, the system could easily be extended to generate a real-time adaption for dynamic scenes. This is also supported by our segmentation of the trajectory. If only few objects in a scene change their position, we can just recompute the segments, which are related to the reference frames of these objects. Otherwise, one would have to recalculate the whole trajectory.

V. Conclusions

Most approaches to robot programming by demonstration require multiple demonstrations to adapt a task to a new situation. Todays one-shot approaches require prior training, or manual post-processing, or are computationally expensive. In this paper, we presented a one-shot method to adapt a demonstrated trajectory to a new situation. Besides the single demonstration, it only requires the scene objects' positions and orientations. The relevant features are selected by a distance-based heuristic. The system segments the demonstrated trajectory and adapts each segment to the new situation. The method is capable of adapting a demonstration to new robot and object positions and orientations. We evaluated the approach qualitatively and quantitatively in various tasks. The approach exactly reproduces a demonstration under unchanged conditions, copying the users demonstrated ground truth. The adaption is fast computable and easily comprehensible by the user, which helps the user to provide suitable demonstrations.

Future work may deal with adaption techniques, that map a sample to more than one reference frame [23] [24]. One could also concentrate on the management of the behaviors. I.e. how the system can recognize and resolve contradictory demonstrations and how the user can easily administrate the adding and removing of behaviors.

Furthermore, the system could be upgraded by an obstacle avoidance. To do so, the transfer frames could be divided into several virtual frames. Then, in reproduction the affected virtual frames could be moved away from the obstacle, bending the trajectory around the obstruction.

References

[1] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," *Robotics Research*, pp. 1–10, 2005.

[2] J. Peters and S. Schaal, "Policy learning for motor skills," *Neural Information Processing*, pp. 233–242, 2008.

[3] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," *2009 Robotics and Automation(ICRA)*, pp. 763–768, May 2009.

[4] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zollner, and M. Bordegoni, "Learning from Human Demonstration and Advice: the machine learning paradigm," *Robotics Research*, vol. 9, pp. 229—-238, 2000.

[5] R. Zollner and M. Pardowitz, "Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration," *Robotics and Automation (ICRA)*, 2005.

[6] R. Zollner, T. Asfour, and R. Dillmann, "Programming by demonstration: dual-arm manipulation tasks for humanoid robots," *Intelligent Robots and Systems (IROS)*, vol. 1, pp. 479–484, 2004.

[7] M. Nicolescu and M. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," *International joint conference on Autonomous agents and multiagent systems*, 2003.

[8] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," *Intelligent Robots and Systems (IROS)*, pp. 5239–5246, Oct. 2012.

[9] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, May 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0921889008001772

[10] D. H. Grollman and O. C. Jenkins, "Incremental learning of subtasks from unsegmented demonstration," *Intelligent Robots and Systems (IROS)*, pp. 261–266, Oct. 2010.

[11] S. Calinon and Z. Li, "Statistical dynamical systems for skills acquisition in humanoids," *Humanoids*, 2012.

[12] T. Cederborg, A. Baranes, and P.-Y. Oudeyer, "Incremental local online Gaussian Mixture Regression for imitation learning of multiple tasks," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010.

[13] A. Wrede, Sebastian Emmerich, Christian Grünberg, Ricarda Nordmann, A. Swadzba, and J. Steil, "A User Study on Kinesthetic Teaching of Redundant Robots in Task and Configuration Space," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 56–81, 2013.

[14] L. Rozo, S. Calinon, D. Caldwell, P. Jim, C. Torras, and I. D. Rob, "Learning Collaborative Impedance-based Robot Behaviors," *AAAI Conference on Artificial Intelligence.*, 2013.

[15] S. Iba, C. Paredis, and P. Khosla, "Interactive multi-modal robot programming," *The international journal of robotics research*, vol. 24, no. 1, pp. 83–104, 2005.

[16] P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso, "Interactive Robot Task Training through Dialog and Demonstration," *Forbes*, 2007.

[17] H. Mayer, I. Nagy, and A. Knoll, "Adaptive control for human-robot skilltransfer: Trajectory planning based on fluid dynamics," *Robotics and Automation*, pp. 10–14, 2007.

[18] Y. Wu and Y. Demiris, "Towards One Shot Learning by imitation for humanoid robots," in *Robotics and Automation (ICRA)*, 2010, pp. 2889–2894.

[19] A. A. N. Kumaar and S. Tsb, "Mobile Robot Programming by Demonstration," *Emerging Trends in Engineering Technology*, vol. 24, no. 4, pp. 206–209, 2011.

[20] M. Kasper, G. Fricke, K. Steuernagel, and E. von Puttkamer, "A behavior-based mobile robot architecture for Learning from Demonstration," *Robotics and Autonomous Systems*, vol. 34, no. 2-3, pp. 153–164, Feb. 2001.

[21] S. Nguyen and P.-Y. Oudeyer, "Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner," *Paladyn*, vol. 3, no. 3, pp. 136–146, 2012.

[22] C. Groth and D. Henrich, "Multi-Tasking of Competing Behaviors on a Robot Manipulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[23] C. Groth and D. Henrich, "One-Shot Robot Programming by Demonstration using an Online Oriented Particles Simulation," in *Robotics and Biomimetics2*, 2014.

[24] C. Groth and D. Henrich, "Single-Shot Learning and Scheduled Execution of Behaviors for a Robotic Manipulator," in *International Symposium on Robotics*, 2014.