Hand Pose-based Task Learning from Visual Observations with Semantic Skill Extraction

Zeju Qiu[1], Thomas Eiband[1,2], Shile Li[1], and Dongheui Lee[1,2]

***Abstract*— Learning from Demonstrations is a promising technique to transfer task knowledge from a user to a robot. We propose a framework for task programming by observing the human hand pose and object locations solely with a depth camera. By extracting skills from the demonstrations, we are able to represent what the robot has learned, generalize to unseen object locations and optimize the robotic execution instead of replaying a non-optimal behavior. A two-staged segmentation algorithm that employs skill template matching via Hidden Markov Models has been developed to extract motion primitives from the demonstration and gives them semantic meanings. In this way, the transfer of task knowledge has been improved from a simple replay of the demonstration towards a semantically annotated, optimized and generalized execution. We evaluated the extraction of a set of skills in simulation and prove that the task execution can be optimized by such means.**

I. Introduction

In traditional robot programming, a human programmer has to manually implement the desired behavior to specify how the robot should act and respond to a certain environmental state. This way of creating robotic applications has some major drawbacks. Firstly, this requires expert knowledge and therefore hinders novice users to intuitively create robot programs. Secondly, even small reconfigurations, for instance environmental changes, might require time-consuming modification of the application. With Learning from Demonstration (LfD) approaches, on the other hand, we can reduce the time-consuming manual programming. In our case, this is solely driven by visually showing the robot a sequence of actions by a teacher. Commonly, a human demonstrator performs single or multiple demonstrations that constitute a data set from which a LfD system tries to find a strategy to reproduce the demonstration [1], [2].

We address the question of how to create a programming paradigm so that even non-experts can create robot applications that reproduce the human behavior. This so-called correspondence problem can be solved by identifying a mapping function between the teacher and learner [1], where the basic concept is shown in Fig. 1. One intuitive way is to use a vision system to observe and learn from the demonstration since we as humans also use this modality to observe and learn new behaviors.

[1] Chair of Human-centered Assistive Robotics, Technical University of Munich (TUM), Munich, Germany

[2] German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Wessling, Germany

{zeju.qiu, thomas.eiband, dhlee}@tum.de, li.shile@mytum.de

This work has been partially supported by Helmholtz Association.

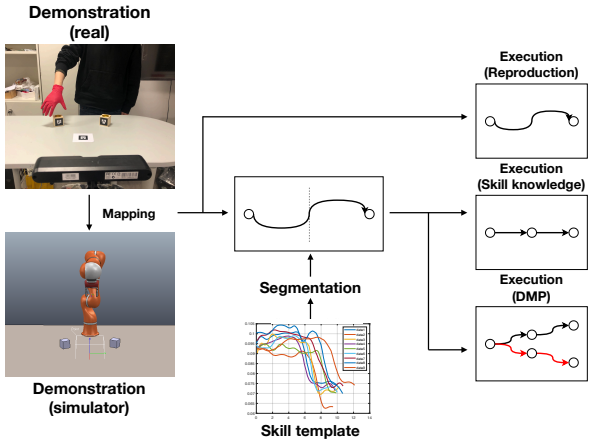

Fig. 1. Framework for learning robot tasks from human demonstrations by extracting skill semantics. The demonstration is tracked by a camera and mapped to the robot execution by considering optimization and generalization to new situations.

Many robot applications have common subroutines, such as pick and place. However, new applications are often costly programmed from scratch. One solution to this problem is task-level programming, where subroutines are used to reduce the programming effort. This requires a segmentation mechanism that separates the demonstration into several actions, so-called skills. Since we define the semantics of a skill, for instance to pick a specific object, we are able to generalize the execution to new object locations. Furthermore, the robotic task becomes interpretable to the user as it can be represented by a sequence of skills. By identifying skill segments from a human demonstration, we can incorporate expert knowledge to optimize the execution, such as removing unnecessary movements from the demonstration. For example, such movements occur during a demonstration when the user does not choose the shortest way while picking an object and we do not want the robot to reproduce them each time.

Our research aim is to extract semantic skills from a hand pose-based demonstration in order to intuitively transfer task knowledge to a robot. An overview of our framework can be found in Fig. 1. We compare different reproduction techniques of the robotic execution in simulation, where the baseline is a simple *replay* of the demonstration. We are able to prove in the experiments that incorporating *skill knowledge* outperforms a simple replay by considering task constraints. Furthermore, by identifying the relationship between manipulation actions and objects, we facilitate Dy-

namic Movement Primitives (*DMP*) [3] as a reproduction method, which allows generalization to new object locations. The main contributions of this work are:

- Proposal of a two-staged segmentation algorithm for manipulation tasks that combines threshold-based segmentation with a probabilistic segment matching.
- Proposal of a set of robotic skills that are parameterized by the demonstration and able to optimize the robotic execution.
- Extraction of such skills directly from hand-pose observation with a semantic sanity check of their sequential order.

II. Related Work

Existing task learning frameworks such as [4]–[6] rely on kinesthetic teaching. In contrast to that, we directly observe the human hand to simplify the task transfer. In [7], grasping tasks have been demonstrated by wearing a data glove that extracts finger joint angles of the human hand pose in combination with a magnetic tracker for the hand base frame. We simplify the technical requirements by just requiring a colored glove that is visually tracked and where the joint positions are extracted from the hand pose estimation algorithm in [8].

Instead of relying on predefined hand gestures as in [9], we extract the task-relevant actions directly from the human demonstration. In [10], a Kinect sensor is used to track the demonstrator's hand movement to achieve teleoperation. Our approach focuses on task learning instead of teleoperation such that the learned task can be autonomously executed even in a changed environment.

We use a deep learning hand pose estimation algorithm that uses a Permutation Equivariant Layer (PEL) with a voting-based scheme [8]. The benefit of this algorithm is that it provides information about fingertips and finger ankles and thus enables our method to analyze the shape of different hand poses during the demonstration. Our mapping function is inspired by [10], where a hand coordinate system is created. However, because of the novel hand-pose estimation algorithm from [8], we can create a more robust coordinate system using our index fingertip, index finger ankle, and thumb tip, instead of the usage of the rough visual separation of the forearm.

Motion segmentation, which separates a long sequence of the human movement into smaller components (movement primitives) by identifying segment points, has long caught the attention of researchers [11] and ranges from simple, online segmentation approaches, e.g. by measuring object proximity and explicit human commands [4], to the probabilistic segmentation of sequences, where consecutive frames belong to different elements of a Gaussian mixture model [12]. In [13], the authors propose an online segmentation algorithm based on velocity features and motion templates in a two-stage recognition process. In the training phase, example data are used to create probabilistic templates and in the following segmentation phase, the observed data are swept for characteristic features that match the pre-defined motion templates. The framing window for the classification is obtained by monitoring zero-velocity crossings (ZVC) and velocity peaks. However, ZVC tends to massively oversegment and creates a large number of false positives. Additionally, their approach requires body-mounted inertial measurement units for the data generation. Our segmentation process has been inspired by [13], but instead of focusing on segmentation and analysis of human motions, our aim is to extract semantically meaningful segments in order to learn an optimized robotic task execution. We are able to make a better initial estimation of the framing window by incorporating distance-based features beside the velocity-based features of [13] into the segmentation process.

A task-level programming paradigm provides the possibility for non-expert users to create robotic applications. Pedersen et al. [14] divide the programming into three layers: the *primitive*, *skill* and *task* layer. The *primitive* layer consists of simple atomic robot movements like a move or close gripper. A *skill* is defined as an object-centered robot behaviour, which incorporates both the sensing and action performed on the object. A *skill* should be parametric in execution, able to estimate the starting condition and able to verify the execution success. A *task* consists of skills and is designed as a complete robot program that is related to a specific goal, for example, assembly or machine tending. Other researchers come up with similar definitions for skills: in [15] the authors proposed a skill portfolio, which is able to solve a large set of industrial tasks. Their proposed skills are: move to, locate, pick up, place, unload, shovel, check, align, open, close, press, release and turn. Similar to [14], our skill formalization is also object-centered. This means a skill is always defined with an object and the spatial relationship to the object can be represented as skill parameter, for example, a grasp pose. In that sense, a skill is interpretable by the human in terms of its name that describes a physical operation, its manipulated objects and meaningful parameters for the execution.

III. Method

Our approach can be divided into three parts: (A) task teaching and data generation, (B) segmentation procedure, and (C) task execution.

A. Task Demonstration and Replay

A new task is demonstrated by the human and recorded with a RGB depth camera. The data of the estimated hand poses and tracked object poses is resampled to the same sampling rate and timestamps because the deep learning-based hand pose estimation algorithm has a lower sampling rate (5 Hz) compared to the object tracking algorithm (30 Hz). A hand coordinate system has been defined (Fig. 2) to enable the control of the robot gripper in the simulator. After filtering and transforming the hand pose into the simulator's reference frame, we can mimic the human hand motions smoothly in the robot simulator.

1) Data processing: A marker attached to the workspace plane serves as the reference frame of the system, where the poses of all the objects and the hand are relative to. Respectively, we define a reference frame in the robot simulation workspace. In the following, we introduce the coordinate frames, with o as object, o' as an object that has been displaced from its initial position, c as camera, h as hand and r as reference coordinate frame. ${}^{r}\boldsymbol{T}_{h}$ denotes the homogeneous transformation from the hand frame to the reference frame with $\boldsymbol{r}$ as the translation and ${}^{r}\boldsymbol{R}_{h}$ as the rotation matrix (Fig. 3).

$$ {}_{r}\boldsymbol{p} = \begin{bmatrix} {}^{r}\boldsymbol{R}_{h} & \boldsymbol{r} \\ \boldsymbol{0}^{\intercal} & 1 \end{bmatrix} \cdot {}_{h}\boldsymbol{p} = {}^{r}\boldsymbol{T}_{h} \cdot {}_{h}\boldsymbol{p} \tag{1} $$

$$ {}^{r}\boldsymbol{T}_{h'} = {}^{r}\boldsymbol{T}_{o} \cdot {}^{o}\boldsymbol{T}_{o'} \cdot {}^{o}\boldsymbol{T}_{h} \tag{2} $$

The data filtering process uses the periodic interpolating cubic spline curve obtained from the given position data, which is a set of fourth-order polynomial equations [16]. The advantage of the spline interpolation is its convenience in the smooth calculation of velocity (and acceleration) and resampling of the data. A fourth-order low-pass Butterworth filter in combination with a two-sided filtering using *Mathworks MATLAB's filtfilt* function has been used to remove noise from the demonstration data with zero phase shift. After preliminary tests, we decided to use a cutoff frequency of 0.0625 Hz for the relatively noisy Euler angles to obtain a smooth robot motion and a cutoff frequency of 0.625 Hz to keep enough details of the characteristic features for template matching (which are independent from the Euler angles).

2) Mapping function: The correspondence problem can be solved by identifying a mapping function between the teacher (demonstrator's hand) and the learner (robot gripper). Therefore, a hand coordinate system is constructed with the positions of the index fingertip (I), the thumb fingertip (T) and the index finger ankle (B), with the origin as the index finger ankle (B) (Fig. 2). The middle point between the index fingertip and the thumb tip corresponds to the tool center point (TCP) of the robot gripper. The z-axis is defined as the line through the origin and the TCP, while the x-axis is normal to the plane containing I and T. Finally, the y-axis is calculated through the cross product of the unit vector x and z. The position of the gripper is defined as equivalent to the TCP in the simulator's reference frame. Accordingly, the gripper orientation is defined as the rotation matrix which rotates the reference coordinate system to our hand coordinate system. The rotation matrix can be obtained with the three orthogonal unit vectors of the hand coordinate system.

B. Human Motion Segmentation

The proposed segmentation procedure is a two-staged segmentation and recognition process: the first step is the segment point modeling, which creates an initial over-segmentation with pre-defined thresholds that outputs a number of data sections (see Fig. 4 top). In the subsequent stage,

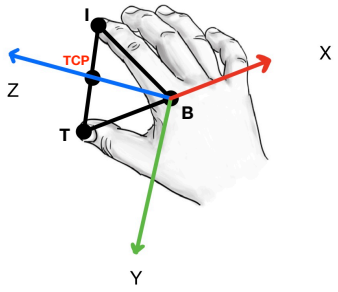

Fig. 2. Hand coordinate system constructed from the three points **I, B, T** (Hand design from [17]).

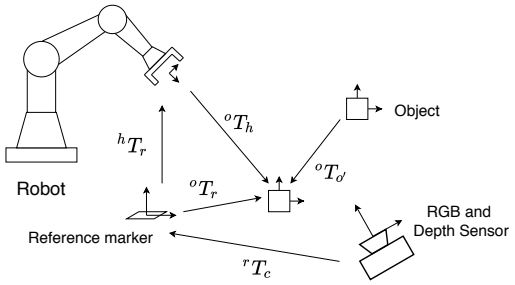

Fig. 3. Coordinate transformations between camera, reference, hand and objects.

template matching is applied to these data sections to reduce the over-segmentation and give the extracted segments semantic meanings in the form of skills (Fig. 4 bottom).

1) Skill definition - Template: In order to classify a new segment into one of the predefined skills, so-called templates have been generated beforehand. In our case, we collected nine demonstrations per skill template, which last between seven to twelve seconds and vary in their velocity, motion and approaching direction. In total, there are seven skills defined: **Pick up**, **Place**, **Locate**, **Stack**, **Push**, **Move** (without object) and **Move with** (with object). Further, eight different features are defined to characterize a skill, as listed in Table I. Six of them are distance-based and two of them velocity-based, with $i, j \in O$, $k \in \{x, y, z\}$, O as the object set, o as the object frame, γ is a small offset, d_i as the object's bounding box dimension, v_{th} as a pre-defined velocity threshold, $\vec{e_z}$ as unit vector z and $\boldsymbol{p}_{\mathrm{i}}$ denotes the position of object i. The value of d_i is pre-defined for each object. After computing the time-series of each feature for each template demonstration, they are encoded in a Hidden Markov Model.

2) Probabilistic Template Encoding: One of the major challenges in the classification of a skill is the temporal and spatial variations between the trained templates and the newly generated data. We chose Dynamic Time Warping (DTW) to eliminate the temporal distorted time series that have similar characteristics but are locally out of phase [18]. Invariance to spatial variations can be achieved by normalizing templates and data points in the interval of $[0, 1]$. For example, different people have different hand sizes and stretches the fingers differently during the demonstration

TABLE I

FEATURE DEFINITION

Feature name	Equation	Threshold
Fingertip distance profile	$\|\boldsymbol{I}(t) - \boldsymbol{T}(t)\|_2$	$d_i + \gamma$
TCP rise/fall profile	$\partial\mathbf{TCP}(t)/\partial z$	0
Object proximity profile	$\|\mathbf{TCP} - \boldsymbol{p}_i\|_2$	$1/2 \cdot d_i + \gamma$
Object distance profile	$\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2$	d_i
Object height profile	$\boldsymbol{p}_i \cdot \vec{e_z}$	$\boldsymbol{p}_i \cdot \vec{e_z} + \gamma$
TCP position in object frame	${}^o\mathbf{TCP} \cdot \vec{e_k}$	$\pm\{1/2 \cdot d_i + \gamma\}$
Object velocity profile	$\|\dot{\boldsymbol{p}}_i\|_2$	v_{th}
Hand velocity profile	$\|\Delta\mathbf{TCP}(t)\|_2$	v_{th}

(e.g. while grasping). Hence, the absolute value of the feature *fingertip distance profile* might vary significantly even though they are executing the same action.

A Hidden Markov Model (HMM) is a statistical model that can be used to encode temporal data and we trained our models with the Baum-Welch algorithm [19]. The so-called forward algorithm returns the probability $P(O|\lambda)$ that an observation sequence O has been generated by a given model λ [19]. Given multiple models, it finds the most likely model that could best reproduce the given observation. By using an HMM, we are able to classify the whole feature time-series belonging to a segment by comparing it with the template models.

In this work, features for each skill are encoded in a HMM model, which is pre-trained and saved. Later on, we want to infer that the observation from a certain feature f belongs to skill s.

3) Stage 1 - Segment Point Modeling: The goal in the segment point modeling process is to obtain the framing window for the first segmentation stage by searching for characteristic features (criteria). The first stage of segmentation divides the observation sequence into data sections and forms the basic structure of the human motion segmentation.

Note that the demonstration data might contain different objects that are tracked. As mentioned in the skill definition, each skill is defined over the hand movement characteristics and the interaction between the hand with a target object, which is called the object-centered definition of the skills. Assume that during a **Place** skill, we want to examine the feature *object proximity profile*. It will be misleading if we falsely calculate the distance between the TCP and a random, irrelevant object, which is not the target object. In this stage, the system analyzes all the objects' velocities during the entire demonstration and labels all the data points with an object number. In the second stage, only the target object data is examined.

A criterion is defined by setting a threshold for a feature (refer Table I). Each time in the demonstration when this threshold has been crossed, a new segmentation point is generated. Preliminary tests revealed that using only single features, e.g. *hand velocity profile*, is insufficient to segment all the skills from each other because they do not explain enough contextual information. The initial assumption was that there is a significant decrease in the velocity while

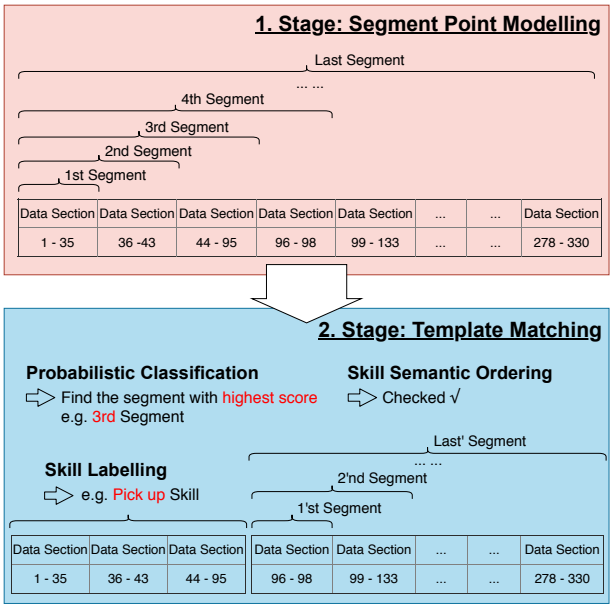

Fig. 4. Two-staged segmentation and skill classification algorithm. 1. Stage (top): cascaded over-segmentation over data sections. 2. Stage (bottom): template matching for skill identification by score of each feature's template with the three major components: probabilistic classification, skill semantic ordering, skill labelling. In this example, the first three data sections have the highest score with the skill **Pick up**. Thus we can label the 3rd segment with the skill name **Pick up**, reassign the remaining data sections to the renumbered segments and repeat the same procedure until all skills has been extracted.

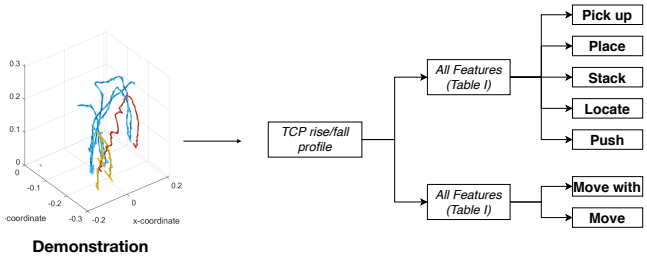

Fig. 5. Decision tree in the skill labelling process. Skills are divided into two groups by considering the feature *TCP rise/fall profile*. Afterwards, all the features are used to label the skill.

demonstrating delicate skills like **Pick up** or **Push** compared to other skills in the demonstration. Instead, the demonstration data unveiled that a human demonstrator can easily switch between two physically coherent skills and therefore only a negligible velocity decrease could be registered.

The first stage of segmentation has been determined by exploiting multiple criteria at the same time. The result is a combination of *fingertip distance profile*, *hand velocity profile* and *TCP rise/fall profile*. This will firstly lead to an over-segmentation, but the number of the segments can be reduced in the next stage with template matching.

4) Stage 2 - Template matching: In the first segmentation stage, we have created several data sections. The goal of the second stage is to reduce the over-segmentation and encapsulate data sections to skill units. The template matching process consists of three major components:

- probabilistic classification
- skill semantic ordering
- skill labelling

We use the forward algorithm to calculate the probability that an HMM with parameters λ_f^s for a skill s generated the observation sequence $\boldsymbol{O}_f^r$ for a segment r and feature f. The observation sequence is the feature's trajectory in its respective data segments and its score is defined by the log-likelihood

$$c_f^{r,s} = \log P(\boldsymbol{O}_f^r | \lambda_f^s). \tag{3}$$

The most likely skill $\hat{s}$ in the skill set S associated to its segment $\hat{r}$ can be found by the maximum final score, given by

$$\hat{s} = \underset{s \in S}{\operatorname{argmax}} \left\{ \max_{r \in R} \{ \sum_{f \in F} c_f^{r,s} \} \right\} \tag{4}$$

$$\hat{r} = \underset{r \in R}{\operatorname{argmax}} \{ \sum_{f \in F} c_f^{r,\hat{s}} \}, \tag{5}$$

where F is the set of all features and R is the set containing all segments r.

a) Probabilistic Classification: We assume that each demonstration can be completely divided into skills from the predefined skill set. The first segment comprises the first data section, the second segment comprises the first and second data section and so forth until the last segment comprises all the data sections, as illustrated in Fig. 4. The algorithm applies the forward algorithm to all the segments and calculates the score of each skill for all the segments. Afterwards, each segment is labeled with the skill that achieves the highest score. In the end, every segment will be compared with each other and the segment with the highest score will be cut off and form the first skill. This process is repeated until the whole observed data has been completely divided into skills (Fig. 4). If the first three data sections combined are more likely than all other combinations of sections (i.e., the 3rd segment has the highest score), as shown in example Fig. 4, these three data sections are merged into one segment and associated with the found skill template (e.g. a **Pick up** skill). We cut the first three data sections off and carry on with the examination of the rest of the data.

b) Skill Semantic Ordering: The idea of the skill semantic order is that a skill sequence should be logically meaningful. For example, the algorithm has discovered a **Place** skill, which implies that there is no object currently holding in hand. If the next recognized skill is a **Place**, this suggests a failure at least in the recognition. The knowledge about the skill semantic order can be used to reduce the computing effort or to evaluate the success of a segmentation. In this paper, the semantic order of skills has been predefined and used to inform the demonstrator about possible failures in the segmentation. In that case, the user should repeat the demonstration to guarantee a solid execution in the simulator. The allowed sequences of different skills can be found in Fig. 6.

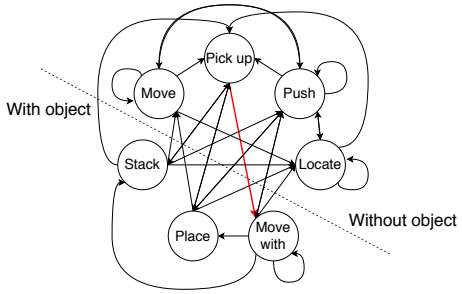

Fig. 6. Skill semantic ordering: an illustration of the sequential relationship between different skills. An arrow represents a possible skill sequence, e.g [Pick up, Move with] (red). Skills are separated into two groups: holding object, not holding object.

c) Skill Labelling: We get one feature score for each HMM model after applying the forward algorithm and the next step is to use this information to compare the likelihood of each skill for a given data segment. We address the question of how to combine the feature scores to calculate the final score for each skill with Eq. (4) and (5). After preliminary tests, we have come to the method to divide our 7 defined skills into two groups. The skills **Move to** and **Move with** are set apart from the other skills and considered as the motion that links the more distinct skills like **Pick up**s. The main reason for this separation is that **Move** and **Move with** have no distinct feature characteristics compared to other skills, indeed they are considered to be connectors between other skills. In our skill definition, the skills **Move to** and **Move with** have the distinct property that the TCP is rising while all the other skills have a constant or decreasing *TCP rise/fall profile*. After examining this feature, some skills will be excluded from the examination and for the rest of the skills, the probabilistic classification will be performed for all features weighted equally (see Fig. 5). The goal is to find the skill $\hat{s}$ with segment $\hat{r}$ from Eq. (4) and (5), which maximizes the final score.

C. Task execution with skill knowledge and DMP

The major advantage of segmentation is the integration of skill knowledge to optimize the execution. We do not need to continuously control the gripper, but rather define the commands for the gripper in the skill definition. For example, for **Pick up** we close the gripper at the end when the object position has been reached. In our scenario, we assume that there exist collision-free paths between the recognized robot actions, which we extract as **Move** or **Move with**. Thus, we can eliminate non-purposeful motion artifacts such as shaky movements from the demonstration by executing the **Move** and **Move with** skills with linear movements. This also lays the foundation for further optimizations by integrating expert knowledge. When a **Pick up** skill is recognized, the robot could optimize for a more stable grasp pose according to its gripper geometry.

By employing DMPs, the robot is able to adapt the execution to new object locations using the recognized skills. A DMP encodes a movement primitive into a second-order,

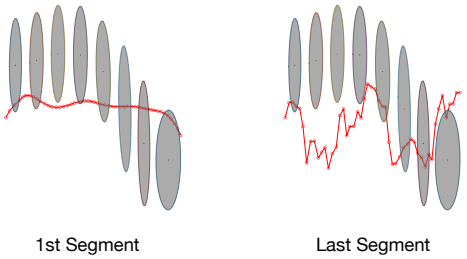

Fig. 7. Template matching between a feature observation sequence (red) and its probabilistic model components (grey) for the skill **Pick up** and the feature *fingertip distance profile* (Experiment 1). Refer to the definition for 1st segment and last segment in Fig. 4.

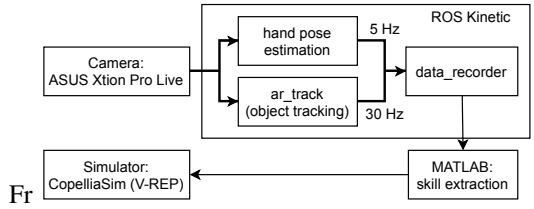

Fig. 8. System overview of the used hardware and ROS packages

non-linear dynamical system with a nonlinear forcing term that can be encoded in radial basis functions [2]. A DMP is denoted as

$$\ddot{\boldsymbol{x}} = k^p(\boldsymbol{\mu}_T - \boldsymbol{x}) - k^v\dot{\boldsymbol{x}} + \boldsymbol{f}(s)\ ,\ \boldsymbol{f}(s) = \sum_{k=1}^{K} \phi_k(s) s \boldsymbol{F}_k \quad (6)$$

with $\boldsymbol{f}(s)$ as forcing term, k^p as stiffness, k^v as damping, $\boldsymbol{\mu}_T$ as goal point, $\phi_k(s)$ as radial basis functions (RBF) and F_K as weights that are learned from the data. The phase variable s is represented by the expression $\dot{s} = -\alpha s$ and encodes the time evolution of the system. The forcing term can be learned from each segment's data and can be used to calculate new trajectories with a new target or starting position. We give a simple example for the DMP usage consisting of two sequentially ordered skills, **Pick up** and **Move with**. After the object has been displaced, the new target grasping pose can be calculated by the relative grasp pose between gripper and object (Eq. (2)), which is a parameter of the skill. The new target grasping pose becomes the new target pose of the **Pick up** motion segment and the new start pose of the **Move with** motion segment. Given this information, we can calculate new motion primitives with DMPs.

IV. Experiments

A. Experimental Setup

We have created several ROS (Robot Operating System) packages to track object and hand poses with an ASUS RGB camera with depth sensor (Fig. 8). With a state of the art hand-pose estimation algorithm [8], a human is able to demonstrate manipulation skills by wearing a hand glove (Fig. 1). Furthermore, the demonstration involves object handling, which is enabled by tracking the object's pose via AR tags [20]. We use the robot simulator *CoppeliaSim* [21] to reproduce the robotic task. Our experiment includes a KUKA LWR 4+ robot, a workspace table and several cuboid-shaped objects. The connection to the simulator is established with its remote API via a socket communication to *Mathworks MATLAB*. Three cuboids equipped with an AR track marker are used to perform the demonstration. Each marker has an individual id and a reference marker (id=0) has been fixed to the table. The camera location can be seen in Fig. 1.

B. Experimental Design

Four experiments with six demonstrations each are defined to test the effectiveness and correctness of the proposed segmentation method. Each experiment shows a different combination of skills. The four experiments are designed to cover all the defined skills and are complex enough to require an effective segmentation strategy. Two experiments have been analyzed in more detail in the experimental results. To validate our segmentation method, we compare the following four segmentation methods:

- proximity-area-based segmentation
- gripper-distance-based segmentation
- ground truth
- **proposed approach**

Hereby, we use the first two methods as baselines. The gripper-distance-based segmentation monitors the fingertip distance and segments if the fingertip distance crosses a threshold. The proximity-area-based segmentation segments if the TCP enters or leaves the proximity region of an object. In the ground truth, an objective human observer watches the demonstration as video and manually annotates the demonstration with skill segments. A meaningful segmentation is given if the result resembles the ground truth segmentation.

C. Results

First, we give an exemplary illustration of the template matching, which is part of the procedure described in Fig. 4. In Fig. 7 we can see the result for the feature *fingertip distance profile* and for two different data segments from Experiment 1. The Gaussian Mixture components of the respective HMM model are depicted with grey ovals. We can see that the first segment (score: 98.3896) matches the model more closely than the last segment (score: -829.8919) (refer Eq. 3).

In Fig. 9 and 10, the results of the segmentation for each experiment are shown. While the baseline methods (Fig. 9 and 10 top) work fine in the first experiment setting, these two methods fail to provide a satisfactory segmentation in the second experiment, because monitoring one feature is not sufficient to extract all the skills. Our segmentation method manages to produce a similar segmentation with the same skill sequence as the ground truth, however, we may notice some temporal variations of the segmentation points compared to the ground truth. This might occasionally cause problems in grasping when simulating the demonstration with extracted skills in the simulator. For example, we do not give explicit human commands to control the gripper

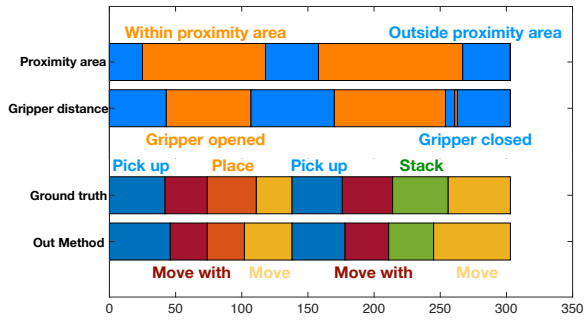

Fig. 9. Experiment 1: Proximity to an object or gripper-finger distance are insufficient to extract skill semantics. Our approach's segmentation matches closely to ground truth observations: [**Pick up**, **Move with**, **Place**, **Move**, **Pick up**, **Move with**, **Stack**, **Move**]. Each color embodies a distinct skill or state.

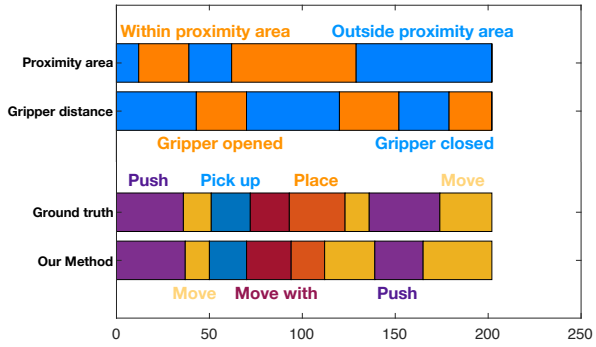

Fig. 10. Experiment 2: Comparison between proximity-based and gripper-distance based segmentation to our approach and ground truth observations: [**Push**, **Move**, **Pick up**, **Move with**, **Place**, **Move**, **Push**, **Move**].

but automate this process with skill knowledge by defining the control of the gripper in the skill definition.

We compared the conditions *replay*, *execution with skill knowledge and DMP* in Fig. 1 and the experimental results can be seen in Fig. 11 and 12. The task flow from each experiment is split into two figures for readability. We can see the system replaces the skills **Move** and **Move with** with linear movements. We notice that in the respective left figure an object has been displaced. The robot is still able to perform the same task with the help of DMP to calculate new trajectories and guarantee a successful execution of the following skills. For example, in the first experiment, the robot returns to the initial ending position after grasping an object from another position and moving with that object. For the skill **Pick up** and **Push** the integration of skill knowledge does not affect the trajectory and therefore the replay curve (solid) overlaps the skill knowledge curve (dotted). In the respective right figure, the remaining skills are illustrated with two curves because object displacement only affects the first two skill segments in this case. The overall path length comparison of all experiments can be seen in Table II. We can observe an overall reduction of the path length after applying the segmentation and skill extraction approach. Path length reduction leads in real-world applications to an increase in efficiency and faster execution.

In Fig. 13 we can see the comparison of the gripper distance (opening of gripper fingers) profile before and after the segmentation. Before the segmentation, we have noisy values of the gripper distance and sending this information directly to the robot gripper would produce task-irrelevant gripper movements. Also, the course of the gripper distance might vary because of unintentional finger movements. With the help of the segmentation, we have the definite information of when to send a command to the robot to close (high to low) or open (low to high) the gripper, which enables stable grasping.

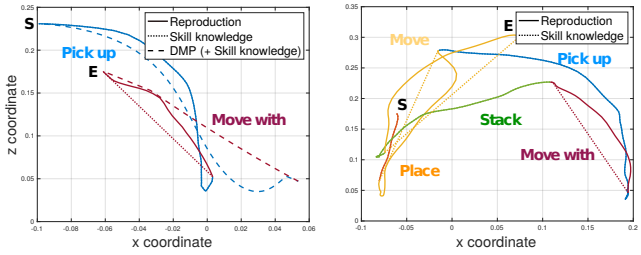

Fig. 11. Experiment 1: Left: a simple replay of the demo (solid) and an execution for a new pick location using a DMP (dashed) for the skills [**Pick up**, **Move with**]. Right: a reproduction of the demo and an execution with skills (dotted) [**Place**, **Move**, **Pick up**, **Move with**, **Stack**, **Move**]. Each color embodies a distinct skill. S denotes the start position and E denotes the end position.

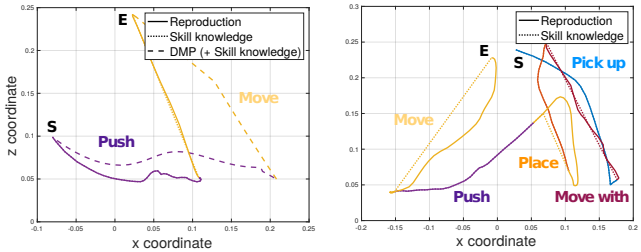

Fig. 12. Experiment 2: Left: a simple replay, skill-based and generalized execution using DMP for skills [**Push**, **Move**]. Right: reproduction and skill-based execution of [**Pick up**, **Move with**, **Place**, **Move**, **Push**, **Move**]. Curve types and color codes are defined same as in Fig. 11.

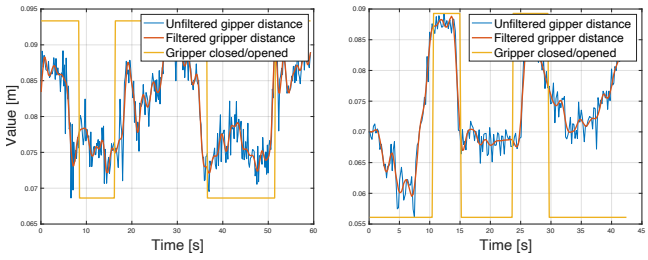

Fig. 13. Gripper distance profile of Experiment 1 (left) and Experiment 2 (right). The blue line represents the unfiltered gripper distance and the red line the filtered gripper distance (low-pass cutoff frequency 0.625 Hz). The yellow line represents the gripper state after the segmentation with high as gripper opened and low as gripper closed.

D. Limitations

A first limitation that has been uncovered through the experiments comes from the recorded demonstration data. The instability in the tracking and recognition of markers and hand poses has been lengthening the demonstration process

TABLE II

PATH LENGTH COMPARISON

	Exp 1	**Exp 2**	**Exp 3**	**Exp 4**
Replay	2.0204	2.1774	1.5876	1.9319
Segmentation	1.8930	2.1187	1.5035	1.6281
Ratio	93.69%	97.30%	94.70%	84.28%

and undermined the intuitiveness of our LfD system. The problem is mainly caused by the physical environment (light condition), but also because of the used algorithms. The hand-pose estimation algorithm functions mostly reliably but puts a ceiling on the possibilities because it was trained without holding an object [8]. The result is that the demonstrator has to always expose all the fingers to the camera during the demonstration to prevent the deformation of the hand. Also, light reflections of the marker could impede the demonstration success. Secondly, the segmentation is sometimes not accurate enough and fails to always produce satisfactory segmentation. The inaccuracy in the segmentation is mainly caused by the framing window. The segments' length might only vary in several data points compared to the next segment, which leads to high similarity and hampers sometimes the recognition procedure. For each skill, 9 demonstrations were recorded to generate a HMM template model because it produces satisfactory results with the given task setting. However, a larger data set of demonstrations will certainly cover more variations of the skills and contributes to a better recognition procedure.

V. CONCLUSION

In this paper, we have presented a LfD system, which observes human demonstrations only by a camera. A robot in the simulator can be taught to reproduce the demonstration with a robust mapping between the robot's gripper and the demonstrator's hand. Further, we propose a segmentation algorithm based on template matching via Hidden Markov Models (HMMs) to segment the human demonstration into semantically meaningful motion primitives, so-called skills. These skills allow optimizing the demonstration by integrating expert knowledge and generalization to new object locations via DMPs.

To further improve the adaptability of our approach we could replace DMP with Task-parameterized Gaussian Mixture Models (TP-GMM) to become more robust in complex environments, e.g. allowing obstacle avoidance. Another research area would be the integration of more skill knowledge to further improve the execution robustness which lays the foundation for more complex tasks. In this paper, we primarily focus on evaluating the feasibility of our approach and verify it with the help of a robot simulator. Therefore, testing on a real robot and evaluating the user's experience would be the next step of our work.

ACKNOWLEDGEMENTS

We thank Alejandro Agostini and Teodora Raicevic for their support on the robotic simulator.

REFERENCES

[1] 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, 2009.

[2] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: a Reference*, P. Vadakkepat and A. Goswami, Eds. Springer, 2019, pp. 1261–1312.

[3] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[4] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction," *Autonomous Robots*, vol. 43, no. 6, pp. 1291–1307, 2019.

[5] F. Steinmetz, V. Nitsch, and F. Stulp, "Intuitive task-level programming by demonstration through semantic skill recognition," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3742–3749, 2019.

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

[7] A. M. Schmidts, D. Lee, and A. Peer, "Imitation learning of human grasping skills from motion and force data," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1002–1007.

[8] S. Li and D. Lee, "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 927–11 936.

[9] H. Deng, Z. Xia, S. Weng, Y. Gan, P. Fang, and J. Xiong, "A motion sensing-based framework for robotic manipulation," *Robotics and biomimetics*, vol. 3, no. 1, p. 23, 2016.

[10] G. Du, P. Zhang, J. Mai, and Z. Li, "Markerless kinect-based hand tracking for robot teleoperation," *International Journal of Advanced Robotic Systems*, vol. 9, no. 2, p. 36, 2012.

[11] J. F.-S. Lin, M. Karg, and D. Kulić, "Movement primitive segmentation for human motion modeling: A framework for analysis," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 3, pp. 325–339, 2016.

[12] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *Proceedings of Graphics Interface 2004*. Citeseer, 2004, pp. 185–194.

[13] J. F.-S. Lin and D. Kulić, "Online segmentation of human motion for automated rehabilitation exercise analysis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 1, pp. 168–180, 2013.

[14] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[15] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, "Does your robot have skills?" in *Proceedings of the 43rd international symposium on robotics*, vol. 6. Verlag, 2012.

[16] T. P. Krauss, L. Shure, and J. Little, *Signal processing toolbox for use with MATLAB®: user's guide*. The MathWorks, 1994.

[17] M. Fussell, "How to draw hands - the ultimate guide," [Online; accessed March 12, 2020]. [Online]. Available: https://thevirtualinstructor.com/how-to-draw-hands.html

[18] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, 2004, pp. 11–22.

[19] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[20] S. Niekum and I. I. Saito, "Ar track alvar ros package," *URL http://wiki. ros. org/ar_ track_alvar*, vol. 3, no. 4, 2017.

[21] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.