©2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Please cite this paper as:

```
@INPROCEEDINGS{Scherzinger2019Contact,
author={S. {Scherzinger} and A. {Roennau} and R. {Dillmann}},
booktitle={2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)},
title={Contact Skill Imitation Learning for Robot-Independent Assembly Programming},
year={2019},
volume={},
number={},
pages={4309-4316},
keywords={},
doi={10.1109/IROS40897.2019.8967523},
ISSN={2153-0858},
month={Nov},}
```

Contact Skill Imitation Learning for Robot-Independent Assembly Programming

Stefan Scherzinger¹, Arne Roennau¹ and Rüdiger Dillmann²

Abstract-Robotic automation is a key driver for the advancement of technology. The skills of human workers, however, are difficult to program and seem currently unmatched by technical systems. In this work we present a data-driven approach to extract and learn robot-independent contact skills from human demonstrations in simulation environments, using a Long Short Term Memory (LSTM) network. Our model learns to generate error-correcting sequences of forces and torques in task space from object-relative motion, which industrial robots carry out through a Cartesian force control scheme on the real setup. This scheme uses forward dynamics computation of a virtually conditioned twin of the manipulator to solve the inverse kinematics problem. We evaluate our methods with an assembly experiment, in which our algorithm handles part tilting and jamming in order to succeed. The results show that the skill is robust towards localization uncertainty in task space and across different joint configurations of the robot. With our approach, non-experts can easily program force-sensitive assembly tasks in a robot-independent way.

I. INTRODUCTION

The robotic automation of assembly tasks is one of the oldest fields of robotic applications and has challenged engineers and scientists ever since. In the domain of robotic assembly, having generalized strategies has been a strong incentive, since it would enable to program many robots in short time by effectively re-using solutions. However, highly varying system dynamics usually require specific strategies to deal with part tilting and jamming during execution, and are difficult to generalize to other robots and work postures. In this work, we propose a data-driven approach to obtain contact skills that encapsulate human manipulation strategies that we extract from demonstrations in simulation. Robots can then execute these skills with a common force-control interface.

Flexible automation and intelligent robots in industry have been considered for decades to be substantial for industrialized countries [1], [2]. Towards this goal, works investigated the mechanics of assembly to derive general analytic solutions and principles, e.g. to handle friction [3], part jamming that arise through sensor imprecisions [4], or targeted planar parts with compliance parameter optimization [5]. Contacts and contact transitions have been investigated between manipulation objects [6], [7], whose semantic information can speed-up the design of compliant motion for robot endeffectors in contact with their environments [8]. Following the task-level approach, works used primitives to compose skills, e.g. by formulating position-force commands, such as 'rotate to level', 'rotate to insert' [9], or concatenating sensorimotor primitives [10], [11], using human inspired recipes [12] or formulating elementary actions in the task frame [13]. The idea of elementary skills has still gained some interest in recent works [14], [15]. However, the ingenuity of finding the primitives that apply best in certain situations and parameterizing them often means a considerable engineering effort, leaving the cognitive performance to the programmer and not to the system. Humans are remarkably skilled workers and join assembly parts with ease, although not necessarily using analytical representations, such as contact states [16]. However, it is not intuitive for us to describe how we deal with tilting and jamming or what strategies we deploy upon getting stuck. Following this insight, human performance can directly be used to obtain skills through imitation learning, such as in *Programming by Demonstration* (PbD)¹ [17], [18], [19], with applications for general object manipulation [20], [21], [22], and industrial assembly processes [23], [24]. Contrary, approaches have shown promising results on contact-rich manipulation tasks without human input [25],[26], also with very tight clearances [27]. Transferring the solutions to arbitrary poses in the workspace or even to other systems, however, requires the training of a new controller, even if initial trajectories are provided [28].

In this work, we aim at developing force-based contact skills to handle jamming and tilting effects that are portable to different robotic manipulators, and can, once learned, serve as robot independent skills for a specific assembly task. To this end, we train a recurrent neural network to learn human-like manipulation strategies from human performance in simulation, and relate those to the relative object's geometry in task space. In contrast to related works, our model predicts sequences of forces and torques, which serve as reference set point for a Cartesian force-control of positioncontrolled robots, which we build upon the idea from an earlier work [29].

The remainder of the paper is as follows: In II we discuss related work and motivate our approach, which we explain in detail in III for our contact skill model, and in IV for the robot-abstracting force-control. V shows our experiments and results. In VI we discuss final aspects and conclude in VII.

¹Stefan Scherzinger and Arne Roennau are with FZI Research Center for Information Technology, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany stefan.scherzinger@fzi.de arne.roennau@fzi.de

²Rüdiger Dillmann is with IAR Institute for Anthropomatics and Robotics, HIS Humanoids and Intelligence Systems Lab, KIT Karlsruhe Institute of Technology, Adenauerring 2, 76131 Karlsruhe, Germany ruediger.dillmann@kit.edu

¹Also Learning from Demonstration (LfD)

II. RELATED WORK

Imitation learning of human assembly skills has been tackled with various approaches, such as using teleoperation [30], [23], [24], teaching with a haptic device [31], direct kinesthetic manipulator guiding [32] or the explicit usage of simulation to record human performance, [33], [34], [35], [36]. Our work incorporates both the idea of teleoperation for human skill extraction and the advantage of simulation to acquire numerous samples in short time, using Long Short-Term Memory (LSTM) [37] as a method to model assembly sequences. A similar approach has been used by Rahmatizadeh et al [38] to learn basic manipulation skills with a gamepad, albeit not explicitly targeting contact skill dominated scenarios or robot transfer.

Early proposals for the usage of simulation are from Ogata and Takahashi [33] and Onda et al [35], [34], in which they used a robotic manipulator as teach device to steer objects in simulation with the aim to extract the visiting of contact states during assembly. They later derived hybrid position/force commands, using the work of Hasegawa et al [9]. Skubic et al [31] learned mappings from sensor forces to contact formations directly, obtaining samples via a haptic device. After classifying sequences of those formations, they commanded the robots with velocities using a finite state machine. In a similar approach, Dong et al [36] relied on performance in a virtual environment to identify the contact states with a Hidden Markov Model (HMM). They used Locally Weighted Regression (LWR) to learn rotation angle correction trajectories for a 3 DOF (Degree of Freedom) task.

Krueger et al [23], and later Savarimuthu et al [24] used magnetic trackers in the assembly objects to record Cartesian trajectories during assembling the Cranfield benchmark set. The robot imitated the human performance in teleoperation mode on a twin of the assembly setup, while recording the forces with the robot's end-effector sensor. Instead of requiring special setup for the skill extraction, Kramberger et al [32] directly guided a light-weight robot during the task, recording Cartesian trajectories and force profiles, which were then generalized, using Locally Weighted Regression (LWR).

In contrast to those works, we steer the assembly objects directly with forces and torques, both during the acquisition of training data and during the execution on the robotic system. There are two main reasons for this choice: First, learning mappings from object's geometry to sequences in wrench space allows us to cope with various disturbances (self-provoked or external), by continuing our work from multiple entrance points. Ideally, we plan to recover from arbitrary, relative object poses. Second: Commanding in wrench space allows us to scale the output of our model easily without changing its semantics. Both require an interface to gravity-compensated force control on the robot as described in section IV.



Fig. 1: Skill extraction. (a) Users solve the task by steering the objects with a teach device, using the simulation's rendered world as feedback for their actions. (b) For a successful assembly, users must correct tilting and jamming of the parts that occur due to friction and small clearances of the setup.

III. MODEL

A. Skill extraction

We consider the task of learning assembly skills from human performance in simulation. To this end, we assume that the geometry of all objects involved in the task is known, and we model the assembly parts as rigid bodies with collision physics in a zero gravity environment as shown in Fig. 1 for an exemplary box assembly. We find that the features mentioned are common place for most robot simulation environments, such as Gazebo [39], which we use for our simulations. Note that our approach does not necessarily require realistic mass nor inertia parameters of the objects, which, in contrast to geometry, could be difficult to obtain. Approximate values are sufficient, as long as steering the objects in simulation feels natural and responsive enough. For the active object, we impose a velocity proportional damping according to

$$\begin{pmatrix} \boldsymbol{I}^{3\times3}d_{\text{lin}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}^{3\times3}d_{\text{rot}} \end{pmatrix} \dot{\boldsymbol{x}} + \boldsymbol{f}^d + \boldsymbol{f}^c = \boldsymbol{0}$$
(1)

in which $\dot{\boldsymbol{x}} = [\dot{x}, \dot{y}, \dot{z}, \dot{r}_x, \dot{r}_y, \dot{r}_z]^T$ is the floating part's 6-dimensional velocity with linear and rotary components, d_{lin} , d_{rot} are linear and rotary damping, $\boldsymbol{f}^d = [f_x, f_y, f_z, t_x, t_y, t_z]^T$ is the wrench of forces and torques applied to its center of mass by the user via the teach device and \boldsymbol{f}^c is a physics engine controlled wrench to maintain contact stability and prevent objects from penetrating.

While applying forces and torques to objects is an essential feat of physics engines, it is commonly difficult to obtain realistic contact forces of those objects with virtual sensors, due to approximations of friction for rigid bodies and numerical constraint instabilities. Our idea was therefore to avoid the dependency of those forces as input features to our neural network: The users correct the outcome of their force-based actions through observing the rendered world in simulation. We did not include any other visual feedback, such as plots of force-torque readings. We assume that the users' demonstrations in this visual servoing approach contain sufficient semantic information to learn error-correcting skills relative to the objects. Through this approach, we aim to prevent depending on force-feedback as input feature that would be prone to suffering the "reality gap" when working on the real system. Still relying on simulation provides us with a method to carry out a big number of trials as easy as possible to generate sufficient data for our deep network.

To this end, users steer the active assembly parts with the help of a teach device, in our case a conventional space mouse, which we deploy as a sensor for 6-dimensional motion. We map those inputs to forces and torques, such that offsets to the sensor's initial position scale linearly with the magnitude of the wrench f^d applied to the object in simulation. We continuously record f^d along with the target pose $\boldsymbol{x} = [x, y, z, q_x, q_y, q_z, q_w]$ of the final assembly operation with the orientation given in quaternion notation, and the objects 6-dimensional velocity \dot{x} . Note that we transform and display all quantities with respect to the moving frame of the active assembly object. The user initiated commands f^d represent expert behavior in each situation, which entail both micro strategies with short time horizon against getting stuck at edges and macro strategies with longer time horizon for more path planning behavior.

B. LSTM-based Contact Skill Models

We use an LSTM-based model (Fig. 2) to learn and generalize human skills in form of *one-to-many* mappings. In contrast to feed forward networks, LSTM cells keep an internal state, enabling them to learn across various time steps. More details can be found in the original work [37] and the refinement [40], which is also the base for the implementation we use. When using them recursively on their own predictions, LSTMs have been shown to produce creative sequences from single inputs [41]. Although residing in another domain, our application has similarities to this approach: In our work, to a given seed input - an estimated state in the middle of an assembly operation - our model should generate a creative sequence that is representative of human behavior in that scenario.

Our idea is to drop the model into a specific situation and let it predict a meaningful sequence of next steps in form of forces and torques f^d . The seed input features are composed as the tuple $[x_0 \dot{x}_0 f_0^d]$, where the subscript $_0$ shall reflect any starting point in time from which on a sequence is to be predicted by our model. We unroll our network over a number of fixed



Fig. 2: Skill model

steps N both during training and inference. The unrolling



Fig. 3: Unrolling the neural network over time.



Fig. 4: Composition of training samples.

procedure is as follows:

$$\begin{aligned} \boldsymbol{y}_{0} &= \boldsymbol{W}_{i} \left[\boldsymbol{x}_{0}, \dot{\boldsymbol{x}}_{0}, \boldsymbol{f}_{0}^{d} \right]^{T} \\ \boldsymbol{y}_{k} &= \text{LSTM}(\boldsymbol{y}_{k-1}), \qquad k \in \{1, .., N\} \\ \boldsymbol{p}_{k} &= \boldsymbol{W}_{o} \boldsymbol{y}_{k}, \qquad k \in \{1, .., N\} \end{aligned}$$
(2)

Note that we use the model's prediction in each step as the next input, as is shown in Fig. 3. In all unrolled steps, the LSTMs share the same parameters, which we optimize together with the input weights W_i and output weights W_o subject to our loss

$$L(\boldsymbol{x}_{0}, \dot{\boldsymbol{x}}_{0}, \boldsymbol{f}_{0}^{d}, \boldsymbol{f}^{d}) = \frac{1}{N} \sum_{k=1}^{N} (\boldsymbol{f}_{k}^{d} - \boldsymbol{p}_{k})^{2}.$$
 (3)

C. Training

Fig. 4 shows the composition of training samples. The records from simulation and readings from the teach device form a multitude of demonstrations, each with an individual temporal length T. The difference in length is due to non-deterministic human performance throughout the task and differences in the random starting poses of the active assembly object. Although the sequences depicted in light blue decode expert behavior, the commands issued by users are not optimal and likely contradictory to certain degree. We assume, however, that they statistically contain sufficient consensus on "the correct behavior" in jamming situations. During training, we take samples randomly from the total of demonstrations. Each sample is comprised of the input seed and the following human performance as sequence of labels. Note that the time span of the sample length Nis smaller than an individual complete demonstration T. We train the network with Back Propagation Through Time (BPTT), using mini-batch stochastic gradient descent.



Fig. 5: Skill model in combination with Cartesian force control. The neural network predicts a sequence of N steps, whose individual elements f^d serve as reference setpoint for the force regulating PD controller. The plant uses realistic manipulator kinematics J, and a virtually conditioned manipulator inertia matrix H.

IV. ROBOT CONTROL

We consider the class of robots with high-gain position servo in the joints, which provide an interface for commanding feed forward joint trajectories $q^d(t)$ with individual reference set points. We further assume that our robots possess sensors to measure the wrench f^s at their endeffectors, i.e. at the grasped assembly objects and that they provide functionality to bias the sensors. This functionality is important for us in order to replicate the zero gravity environment we had chosen during the skill extraction in simulation. Relying on additional sensors is not restrictive, given that 6-axis force-torque sensors have become widely available for industrial robots. To derive our control concept, we start with the common equations of motion for an articulated rigid body system in joint coordinates q

$$\boldsymbol{H}\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}} + \boldsymbol{C}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) = \boldsymbol{\tau}_{u} + \boldsymbol{J}^{T}\boldsymbol{f}^{ext} \tag{4}$$

where H(q) denotes the positive definite joint inertia matrix, C encompasses centrifugal and Coriolis terms, as well as gravitational forces, and τ_u denote the motor torques in the joints of the manipulator. External forces and torques f^{ext} , acting on the end effector map to joint space with the manipulator Jacobian J(q). We skip the dependency of qin further notation for brevity. Under the assumption that we move slowly in contacts and that the robots we consider already compensate for gravity in their lower level control, we omit C and τ , such that we obtain

$$\ddot{q} = \boldsymbol{H}^{-1} \boldsymbol{J}^T \boldsymbol{f}^{ext} \tag{5}$$

as an instruction to move according to external disturbances. In contrast to our previous work [29] we use the approximation of (5) for the quasi static case. This mapping computes the instantaneous joint acceleration of our virtual system, reacting in direction of our wrench vector f^d . While Jreflects our real system's kinematics, we chose H by setting the masses of all links to some unit values. Note that we do not estimate the real system dynamics. Instead, we follow the idea from [29], and deploy (5) as a *forward dynamics solver* for the inverse kinematics problem on a virtually conditioned twin of the real manipulator. After double time integration we obtain the new joint commands q^d , which we sent openloop to the black box control of the robot, whose low-level joint position servo compensates for dynamics-introduced



Fig. 6: Coordinate systems and transformations to describe task and robot relations.

disturbances and gravity. The force-torque sensor's readings close the control loop in contacts with the environment. We discuss limitations of this black box restriction briefly in section VI.

Fig. 5 depicts the control scheme, in which f^d is the feed forward prediction of the neural network at each time step and q^d is the joint position set point for the robot. The *PD* controller regulates f^c according to $f^c = k_p e + k_d \dot{e}$ for each of the six Cartesian components until f^d is equilibrated by the individual measured components in f^s . The gains k_p and k_d are, in combination with the chosen dynamics of H, a partly redundant means to adjust the systems responsiveness. Our overall incentive is to achieve that the assembly objects respond equally to f^d , both during supervised training in simulation (without manipulator) and during neural network controlled execution on real robotic manipulators.

We compute the input features for the neural network using joint state feedback q, \dot{q} from the robot. Fig. 6 shows the reference frames and relationships. In the following notation, the sub- and superscripts t, e and b stand for *target, end-effector* and *base* respectively. We assume that an estimation of the final target pose $\{t\}$ of the assembly operation is given with respect to the robot base $\{b\}$, which we denote with the homogeneous transformation ${}^{b}T_{t}$. Its ground truth will coincide with the robot's end effector pose ${}^{b}T_{e}$ after a successful task execution. The current endeffector pose is computed according to ${}^{b}T_{e} = g(q)$ at every time step, using the position sensors readings q and a *forward kinematics routine g*, which we do not further specify. Using the inverse of this transform $({}^{b}T_{e})^{-1} = {}^{e}T_{b}$, the estimated target pose can be formulated with respect to the moving end-

TABLE I: Specifications of the assembly task setup

cube	side plates		mockup clearance	
edge length	thickness	diameter	trans.	rot.
[mm]	[mm]	[mm]	[mm]	[deg]
400	4	145	2	0.8

effector frame according to ${}^{e}T_{t} = {}^{e}T_{b}{}^{b}T_{t}$, from which it is straight-forward to extract our input features in quaternion notation. Additionally, we compute the current end-effector velocity $J\dot{q}$, and likewise display the solution seen from the end effector frame, using ${}^{e}R_{b}$ as the pure rotational part of ${}^{e}T_{b}$.

Deriving the input features for our neural network summarizes as follows:

$$[x, y, z, q_x, q_y, q_z, q_w]^T \leftarrow^e \boldsymbol{T}_t [\dot{x}, \dot{y}, \dot{z}, \dot{r}_x, \dot{r}_y, \dot{r}_z]^T \leftarrow \begin{pmatrix}^e \boldsymbol{R}_b & \boldsymbol{0} \\ \boldsymbol{0} & e \boldsymbol{R}_b \end{pmatrix} \boldsymbol{J} \dot{\boldsymbol{q}}$$
(6)
$$[f_x, f_y, f_z, t_x, t_y, t_z]^T \leftarrow \boldsymbol{f}^d$$

V. EXPERIMENTAL RESULTS

We conducted a set of experiments to evaluate our approach of contact skill imitation learning in simulation with transfer to a real robot.

A. Implementation and setup

We implemented our neural network model in Python, using the machine learning framework Tensorflow [42], and implemented the force control from Fig. 5 in C++ as a realtime ROS-controller for the ROS-control framework [43]. We designed the test setup to include spots for form-closure effects, such as collisions and jamming during insertion. We realized this through a set of cube structures with round plates on the sides, as illustrated in Fig. 7. Table I summarizes the details of the task setup.

We further chose the Universal Robots UR10 as exemplary platform for the task transfer, which is a common place, joint position-controlled, industrial robot. It provided both the reach and the end-effector load capacity for our task.

B. Contact skill learning

The training samples for the task of our experiments were generated in the simulation environment as described in III-A, using a logging rate of 100 Hz. We trained on approximately 1000 demonstrations, each representing a successful insertion by a human expert, classically lasting between 10 s and 15 s, depending on the random starting poses. For all experiments we used 50 cells in our LSTM layer, a minibatch size of 512 and sequences of N = 50 steps in BPTT, corresponding to time slices of 0.5 s in the training set. We applied Dropout [44] as regularization technique to prevent our network from overfitting, and used Adam [45] as optimizer.

We tested the learned skills with bringing the neural network in a multitude of unseen jamming situations in simulation and letting it solve the task. Five of these starting



Fig. 7: Starting poses for the experiments in simulation. No manipulator is involved in these experiments.



Fig. 8: Absolute forces applied by our model along the way to the goal pose. The curves a - e correspond to the starting poses of Fig. 7 $\,$

poses are depicted in Fig. 7. The purple cube was oriented in such a way that it was mandatory for the model to retreat and to apply a more sophisticated strategy than simple goal-directed pushes to succeed. We set the serving rate of the neural network to 50 Hz, covering a time span of 1s in simulation. Fig.s 8 and 9 show the course of the forces and torques as applied by our model along the Euclidean distance to each goal position (reading from right to left). We found that the forces rapidly increased at the beginning of the assembly task, and then characteristically dropped at approximately 0.25 m, which indicates that our model has learned the correspondence of this part of the geometry with the difficult initial insertion phase. This hurdle is located where the edges of the active assembly cube collide with the side plates in the mockup. The according plot of torques underlines this assumption: After correcting the initial orientation, the torques peak again where the forces drop. This shows the neural network's effort to get the insertion right at this point, which it had generalized from the human demonstrations. Only when successfully passing this bottleneck was the network more likely to apply forces of a higher magnitude, which nearly vanish along with the torques upon reaching the goal position. The results show that our model has learned from human demonstrations to cope with jamming situations for this specific task, and linking its predictions to the relative objects' geometry.

C. Task to robot transfer

In this experiment, our goal was to transfer the learned skill to actual hardware and evaluate its robustness with respect to an imprecise target. To this end, we corrupted our ground truth ${}^{b}\mathbf{T}_{t}$ of the final assembly with a random offset in the margin of 50 mm linear displacement and 5 deg rotational displacement, which we applied along random directions in a multitude of trials. Figure 10 shows an excerpt



Fig. 9: Absolute torques applied by our model. The curves a - e correspond to the starting poses of Fig. 7

of the trials in different joint configurations for the robot. During the runs we let our neural network predict sequences of 50 steps, which we executed on the robot with our force control during a constant time window of 2.5 s for each sequence. The force control was running at 125 Hz (0.008 s cycle time, lower limit for the UR10), obtaining updates f^d to the force regulator every 0.05 s. For this experiment we constantly scaled the model's force output with a factor of 1.5 and the torques with 2.0 respectively. We determined the magnitude in test runs prior to starting the evaluation. Note that the ability to scale our model in this way provided an easy and fast mechanism to fine-tune our skill to the real hardware.

Fig. 11 shows the cumulative histogram (right to left) of the trials vs. distance to the goal (ground truth). We discontinued the individual executions when our skill model either made no further progress or when we successfully reached the goal pose. The curve shows that our model frequently got stuck at approximately 250 mm to the goal position, which relates to the decisive region, as was also experienced with the forces and torques from Fig. 8 and Fig. 9 respectively. Note that all tries except for two were successful after surpassing this region, as indicated by the almost horizontal line of Fig. 11. Fig. 12 shows the performance of our model for random error combinations. The blue circles represent successful executions. As long as the errors do not exceed certain ranges, the results indicate a robust performance for up to five times the linear clearance and three times the rotational clearance of the task. For further increased errors. our model still succeeded in some cases, which could be due to the random error direction being sometimes less severe. The results show that our skill model for this task, although purely trained on simulated data, was able to solve the task on the real hardware, using a robot that was not part of the learning process.

Videos are included in the supplementary material.

VI. DISCUSSION

A. Robot control

During the experiments on the real platform we anticipated that singularities of the robot kinematics decreased the performance of the force controller. This would also be the case for robots with limited morphology. We propose to compute the kinematic manipulability in the workspace prior to executing the task and position the robot accordingly. Additionally, we consider articulated robots with a minimum of 6-axis for this purpose. We also observed that the forcetorque predictions of our model were partly compensated by measurements of the force/torque sensor. This was especially the case when the robot end-effector had rotated strongly from its taring position. If sufficiently known, the masses and inertias of the active assembly object could be gravity compensated during runtime for further improvements.

A control rate of $125 \,\mathrm{Hz}$ is commonly slow for stiff contacts. To maintain contact stability with our setup, we had to execute the task at low speeds. Robots with faster readwrite cycles might therefore be more suitable for productive applications.

B. Skills

The time span of our network's memory covered only few seconds during training and inference. Yet, we observed that the robot's execution on the setup followed several longer-term strategies in the neighborhood of tens of seconds. We address this effect to the chaining of small-scale, local sequences from the model's predictions that together form strategies that are more complex over the course of execution. By setting the friction conservatively high in simulation, we motivated the human experts to avoid scratching along surfaces (because it slowed them down), and instead to exploit clearance where possible. Our results show that the demonstrations in simulation contained sufficient contactgeometry semantics to solve our real-world task, such that our model could overcome friction on the real setup without including force-torque measurements in its input features.

C. Scalability of the approach

Although input feature scaling in our network provides a basic generalization for the size of the objects, we assume limited transferability to objects with highly different shapes. This is due to the implicit encoding of geometry in the object-relative poses. However, reusing the network's weights in combination with fine tuning for another task is a promising direction for further research.

Finally, we want to mention that obtaining training data with the real robot in teleoperation is also feasible with our approach, but implies more effort for human experts in obtaining a similar number of labeled executions, albeit then including real task parameters. Note that although 1000 samples seem much, this corresponds to approximately three hours in simulation for obtaining a robot-transferable skill without requiring expertise in robot programming.

VII. CONCLUSIONS

We presented a robot independent method to extract and learn object-relative contact skills from human demonstrations in simulation, using an LSTM-based neural network. Our model learned to correct part tilting and jamming in contacts, which we evaluated on a cube assembly task on real hardware for different joint configurations. Although



Fig. 10: Skill execution on the UR10. Our experiments included 227 trials with randomly corrupted target poses.



Fig. 11: Cumulative histogram of trials, summing-up from right to left.



Fig. 12: Performance of the skill with respect to localization errors of the setup. The target poses were corrupted randomly. The clearance between objects was 2 mm and 0.8 grad. The three classes categorize the individual executions, depending on distance to the ground truth goal pose. Blue circles represent successful trials.

solely trained on simulated data, the obtained skills were carried out robustly by a Universal Robots UR10, which had not been included in the training process. Excluding real force-torque sensor measurements from the network's input features helped this transfer. An advantage of our method is the ability to scale the neural network's predictions without loosing their semantics: Users can deploy their skills with reduced magnitudes for safe test runs.

ACKNOWLEDGMENT

We would like to thank Christoph Ganser for building the hardware demonstrator. This work was supported in part by the German Aerospace Center (DLR), under national registration no. 50RA1503, and the German Federal Ministry of Education and Research (BMBF) under funding code 16SV7715.

REFERENCES

- Nitzan and Rosen. "Programmable Industrial Automation". In: *IEEE Transactions on Computers* C-25.12 (Dec. 1976), pp. 1259–1270.
- [2] D. Nitzan. "Development of intelligent robots: Achievements and issues". In: *IEEE Journal on Robotics and Automation* 1.1 (Mar. 1985), pp. 3–13.
- [3] J. M. Schimmels and M. A. Peshkin. "The space of admittance control laws that guarantees force-assembly with friction". In: *IEEE International Conference on Robotics* and Automation (ICRA). 1993, pp. 443–448.
- [4] Daniel E Whitney. "Quasi-static assembly of compliantly supported rigid parts". In: *Journal of Dynamic Systems, Measurement, and Control* 104.1 (1982), pp. 65–77.
- [5] A. Stemmer, A. Albu-Schaffer, and G. Hirzinger. "An Analytical Method for the Planning of Robust Assembly Tasks of Complex Shaped Planar Parts". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Apr. 2007, pp. 317–323.
- [6] Jing Xiao and Xuerong Ji. "Automatic generation of highlevel contact state space". In: *The International Journal of Robotics Research* 20.7 (2001), pp. 584–606.
- Peng Tang and Jing Xiao. "Automatic generation of contact state graphs based on curvature monotonic segmentation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2006, pp. 2633–2640.
- [8] Wim Meeussen et al. "Contact-state segmentation using particle filters for programming by human demonstration in compliant-motion tasks". In: *IEEE Transactions on Robotics* 23.2 (2007), pp. 218–231.
- [9] T. Hasegawa, T. Suehiro, and K. Takase. "A model-based manipulation system with skill-based execution". In: *IEEE Transactions on Robotics and Automation* 8.5 (Oct. 1992), pp. 535–544.
- [10] J. D. Morrow and P. K. Khosla. "Sensorimotor primitives for robotic assembly skills". In: *IEEE International Conference* on Robotics and Automation (ICRA). Vol. 2. 1995, pp. 1894– 1899.
- [11] J. D. Morrow and P. K. Khosla. "Manipulation task primitives for composing robot skills". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 1997, pp. 3354–3359.

- [12] W.S. Newman et al. "Force-responsive robotic assembly of transmission components". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. 1999, 2096–2102 vol.3.
- [13] H. Bruyninckx and J. De Schutter. "Specification of forcecontrolled actions in the "task frame formalism"-a synthesis". In: *IEEE Transactions on Robotics and Automation* 12.4 (Aug. 1996), pp. 581–589.
- [14] A. Wahrburg et al. "Combined pose-wrench and state machine representation for modeling Robotic Assembly Skills". In: *IEEE/RSJ International Conference on Intelligent Robots* and Systems (IROS). 2015, pp. 852–857.
- [15] L. Halt et al. "Intuitive Constraint-Based Robot Programming for Robotic Assembly Tasks". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 520–526.
- [16] Ellen Klingbeil, Samir Menon, and Oussama Khatib. "Experimental Analysis of Human Control Strategies in Contact Manipulation Tasks". In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 275–286.
- [17] Rüdiger Dillmann, M Kaiser, and Ales Ude. "Acquisition of elementary robot skills from human demonstration". In: *International Symposium on Intelligent Robotics Systems*. Citeseer. 1995, pp. 185–192.
- [18] C. P. Tung and A. C. Kak. "Automatic learning of assembly tasks using a DataGlove system". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1995, pp. 1–8.
- [19] Rüdiger Dillmann. "Teaching and learning of robot tasks via observation of human performance". In: *Robotics and Autonomous Systems* 47.2 (2004), pp. 109–116.
- [20] Rainer Jäkel et al. "Representation and constrained planning of manipulation strategies in the context of programming by demonstration". In: 2010, pp. 162–169.
- [21] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input". In: Advanced Robotics 25.5 (2011), pp. 581–603.
- [22] Leonel Rozo, Pablo Jiménez, and Carme Torras. "A robot learning from demonstration framework to perform forcebased manipulation tasks". In: *Intelligent service robotics* 6.1 (2013), pp. 33–51.
- [23] Norbert Krüger et al. "Technologies for the Fast Set-Up of Automated Assembly Processes". In: KI-Künstliche Intelligenz 28.4 (2014), pp. 305–313.
- [24] T. R. Savarimuthu et al. "Teaching a Robot the Semantics of Assembly Tasks". In: *IEEE Transactions on Systems, Man,* and Cybernetics: Systems PP.99 (2017), pp. 1–23.
- [25] Levine, N. Wagener, and P. Abbeel. "Learning contactrich manipulation skills with guided policy search". In: *IEEE International Conference on Robotics and Automation* (*ICRA*). 2015, pp. 156–163.
- [26] Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.
- [27] T. Inoue et al. "Deep reinforcement learning for high precision assembly tasks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 819–825.
- [28] G. Thomas et al. "Learning Robotic Assembly from CAD". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–9.
- [29] S. Scherzinger, A. Roennau, and R. Dillmann. "Forward Dynamics Compliance Control (FDCC): A new approach

to cartesian compliance for robotic manipulators". In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2017 pp. 4568-4575

- and Systems (IROS). 2017, pp. 4568–4575.
 [30] Jie Yang, Yangsheng Xu, and C. S. Chen. "Hidden Markov model approach to skill learning and its application to telerobotics". In: *IEEE Transactions on Robotics and Automation* 10.5 (Oct. 1994), pp. 621–631.
- [31] M. Skubic and R. A. Volz. "Acquiring robust, forcebased assembly skills from human demonstration". In: *IEEE Transactions on Robotics and Automation* 16.6 (Dec. 2000), pp. 772–781.
- [32] Aljaž Kramberger et al. "Transfer of contact skills to new environmental conditions". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2016, pp. 668–675.
- [33] H. Ogata and T. Takahashi. "Robotic assembly operation teaching in a virtual environment". In: *IEEE Transactions* on Robotics and Automation 10.3 (June 1994), pp. 391–399.
- [34] H. Onda et al. "Assembly motion teaching system using position/force simulator-generating control program". In: *IEEE/RSJ International Conference on Intelligent Robots* and Systems (IROS). 1997, pp. 938–945.
- [35] H. Onda, H. Hirukawa, and K. Takase. "Assembly motion teaching system using position/force simulator - extracting a sequence of contact state transition". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1995, pp. 9–16.
- [36] Shen Dong and Fazel Naghdy. "Application of hidden Markov model to acquisition of manipulation skills from haptic rendered virtual environment". In: *Robotics and Computer-Integrated Manufacturing* 23.3 (2007), pp. 351– 360.
- [37] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735– 1780.
- [38] Rouhollah Rahmatizadeh et al. "From virtual demonstration to real-world manipulation using LSTM and MDN". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [39] Nathan P Koenig and Andrew Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2004, pp. 2149–2154.
- [40] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12 (2000), pp. 2451–2471.
- [41] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164.
- [42] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015.
- [43] Sachin Chitta et al. "ros_control: A generic and simple control framework for ROS". In: *The Journal of Open Source Software* (2017).
- [44] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.
- [45] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).