Learning by Demonstration and Robust Control of Dexterous In-Hand Robotic Manipulation Skills

Gokhan Solak[1] and Lorenzo Jamone[1]

***Abstract*— Dexterous robotic manipulation of unknown objects can open the way to novel tasks and applications of robots in semi-structured and unstructured settings, from advanced industrial manufacturing to exploration of harsh environments. However, it is challenging for at least three reasons: the desired motion of the object might be too complex to be described analytically, precise models of the manipulated objects are not available, the controller should simultaneously ensure both a robust grasp and an effective in-hand motion. To solve these issues we propose to learn in-hand robotic manipulation tasks from human demonstrations, using Dynamical Movement Primitives (DMPs), and to reproduce them with a robust compliant controller based on the Virtual Springs Framework (VSF), that employs real-time feedback of the contact forces measured on the robot fingertips. With this solution, the generalization capabilities of DMPs can be transferred successfully to the dexterous in-hand manipulation problem: we demonstrate this by presenting real-world experiments of in-hand translation and rotation of unknown objects.**

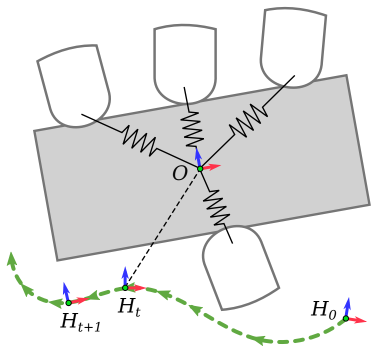
I. INTRODUCTION

Hand manipulation is a crucial skill for the robots that will work together with humans and use objects that are designed for human hands. Complex human-like robot hands can accomplish a large variety of tasks which would otherwise require special-purpose end-effectors. However, some of these tasks are hard to program manually, as they have complex trajectories (i.e., writing) or have significant acceleration profiles (i.e., cutting). In addition, the robust control of these actions is not trivial, since it requires the coordination of multiple articulated robots (i.e. the fingers) which have to exert appropriate forces on the manipulated object to ensure both grasp stability and the desired in-hand motion.

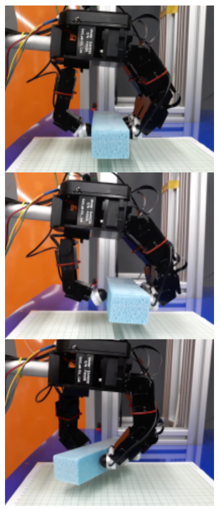
Robot learning can be highly beneficial to achieve effective control of complex actions since less prior knowledge about the world (e.g. object properties) and about the task (e.g. motion trajectories) are required. This can be framed as either autonomous robot learning (e.g. model learning [10] or reinforcement learning [6]) or learning from human demonstration [1]; interestingly, the two approaches can also be combined, as in recent reinforcement learning results [11]. Here we follow the learning by demonstration approach, since it does not require to collect large amounts of data, and in particular the Dynamical Movement Primitives (DMP) framework [4] which allows to learn a motion from a

*This work was partially supported by the EPSRC UK (with projects MAN^3, EP/S00453X/1, and NCNR, EP/R02572X/1).

[1]Gokhan Solak and Lorenzo Jamone are with ARQ (Advanced Robotics at Queen Mary), School of Electronic Engineering and Computer Science, Queen Mary University of London, UK {g.solak,l.jamone}@qmul.ac.uk



(a)



(b)

Fig. 1. (a) Virtual spring framework and object trajectory control. Robot fingertips are connected to the object frame O with virtual springs. Object pose is controlled in order to minimize the error between O and the reference frame H_t. The reference frame follows a trajectory that is learned from demonstration. (b) Our experiment setup. The initial state (top image), the final state after translation (middle image) and the final state after rotation (bottom image).

single demonstration and to generalize it to different conditions (e.g. initial/final pose of the object, duration of the movement). However, the application of DMPs to in-hand manipulation is not trivial, and to our knowledge has not been attempted so far, as it requires the coordinated control of the trajectory of multiple fingers, which have to maintain contacts and to exert appropriate forces on the manipulated object simultaneously. In order to satisfy these requirements, we use a compliant controller, based on the virtual springs framework (VSF [21], [2], [9]), which takes advantage from real-time force feedback provided by force sensors mounted on the robot's fingertips.

Overall, our solution combines DMPs, VSF and force feedback to achieve effective in-hand manipulation of unknown objects: while DMPs permit to extract the desired object trajectories from a single human demonstration, and to easily generalize them to new conditions, the VSF with force feedback allows to keep coordination between the fingers and proper contact forces with the object, without requiring an explicit model of the manipulated object (Fig. 1.a).

We demonstrate the effectiveness of our system through real-world experiments, using a four fingers Allegro Hand robot with 16-DOFs, equipped with OptoForce 3D force sensors on the fingertips and mounted on a 6-DOFs UR5 robot arm, autonomously performing in-hand translations and rotations of an object (Fig. 1.b). The robot learns the task from a single demonstration on an object and reproduces it with different initial state, final state, object size conditions.

The rest of the paper is organized as follows. In Section II we review the related works on these methods and other approaches to our problem. Details of our system are given in Section III. The experiment setup and results are presented in Section IV. Then, in Section V, we summarize our conclusions and sketch future work directions.

II. RELATED WORKS

Learning from human demonstration is based on transferring a human expert's knowledge to a robot. This approach proceeds as one or more experts demonstrate the desired task and a learning method is used to create a control policy. Representation of learned control knowledge is important for the generality and robustness of the method [1]. Expressing the policy in a parametrized form to account for task-level adjustments is fundamental. Task conditions may change the initial state, goal state, object parameters, environment constraints or timing. The representation should also account for the requirements of real-world control problems such as continuous action/state domains, robustness, and stability. *Dynamical systems* are widely used to represent control policies, and existing formulations allow generalization to different task conditions. There are various frameworks built on dynamical systems such as dynamical movement primitives (DMP) [4] and stable estimator of dynamical systems (SEDS) [5]. Although principally similar, these frameworks differ in some aspects. The most prominent ones of these differences are that in SEDS dynamical system is time-invariant and multi-dimensional while in DMP there is a temporal phase variable and each dimension is modeled with an independent system.

In this work, DMP is chosen as it is more mature, with many extensions been proposed over time to increase its generalization capabilities. However, SEDS remains as a possible path to explore.

Although DMP is a well-established framework for learning robot control from demonstration, it is not adequately applied for dexterous hand manipulation yet. DMP is usually applied to object manipulation problems where a single manipulator is used [4], [7]. However, dexterous hand manipulation requires coordinated behavior of multiple manipulators, i.e. fingers. In this case, dynamical systems controlling the fingers cannot be independent of each other. A coupling should be defined between the DMPs in order to achieve coordinated behavior. This coupling is needed for both robustness to perturbations that can happen on some of the fingers and generalization of the initial and goal states where spatial relations of fingers matter. Some existing works apply coupling for coordination in bimanual manipulation tasks ([3], [18]). Both approaches modify the DMP formulation by adding a coupling term that affects the behavior of the second manipulator. Another related approach [13] based on SEDS representation, uses the state of one manipulator as the phase of another to achieve coupling. In their application, the evolution of hand system is driven by the arm system. When the master system, i.e. arm state is perturbed, the slave system, i.e. hand state is set back accordingly. [14] use a specific formulation to obtain generalizable policies, named Task Parametrized Gaussian Mixture Model (TP-GMM). Task-specific Cartesian frames are encoded in TP-GMM to adjust the trajectory. They encode express the Cartesian frame of the secondary manipulator in the primary manipulator's frame to realize coupling in a bimanual task. Bimanual manipulation methods are related to our problem, however, they cannot be applied directly on dexterous manipulation as they are not limited to point contacts, i.e. the object can be tightly grasped by manipulators.

In this work, we use a virtual springs approach for the coordination between multiple fingers. It has been already applied to grasping [2] and dexterous manipulation problems ([21], [9]). In our approach, we do not modify the DMP formulation, but create a coupling at a lower level of control. Virtual springs also add compliance to the system as shown in [2]. This choice also allows us to learn object-centric trajectories, hence the task-level control is independent of the underlying robot hardware.

In addition, we use the real-time measurements of the contact forces at the fingertips to improve grasp robustness and to prevent the object from slipping. Similar approaches in the literature have used tactile sensors to estimate the contact force and to compute corrective control terms [12]; in other cases, tactile sensors or force sensors have been used to detect [19] or to anticipate [20] a slip, before applying the corrective control term. [8] is similar to our work as they use virtual springs for grasping and use tactile information for adapting the grasp. However, they adapt the grasp in a static scenario when the grasp is predicted to be unstable, while we continuously adapt the stiffness of the virtual springs in real-time; this allows to keep a stable grasp of the object while performing the in-hand manipulation.

III. METHODOLOGY

We combine DMPs and virtual springs at different levels. While DMPs are used to learn the motion of the virtual object in Cartesian-space, virtual springs are used to control the fingertip forces to keep the object in grasp and create a compliant coupling between fingers. Thus, we can say that DMPs are located at a higher-level, i.e. task-level. Learned DMPs can be used to reproduce trajectories for different task conditions, such as the execution time, the initial and the goal states. These task trajectories are transformed into fingertip frame, then applied inverse kinematics to obtain joint-space trajectories of all fingers. Finally, the joint-space trajectories are executed using PID control. Virtual springs appear at this level, they generate the fingertip forces which are transformed to joint-torques using the manipulator Jacobian. Fingertip forces ensure contact with the object. Stiffness of finger-object springs are updated continually to keep the fingertip force magnitude in a boundary. Force magnitude is monitored using sensors.

Demonstrations are recorded kinesthetically, that is, the supervisor holds the robot physically and guides it to achieve the task at hand as shown in Figure 3.a. We activate the virtual springs to maintain the grasp while recording the demonstration. This helps the supervisor to focus on the object motion. Kinesthetic teaching eliminates the correspondence problem [1], however, it also limits the dexterity of the motion as it is difficult to move multiple fingers simultaneously.

A learned action is stored as a set of DMPs. A DMP is trained for each dimension of the motion. Since we learn object trajectories in task space, a state contains the pose of the object. The pose is encoded by 9 values: 3 for the Cartesian position and 6 for the rotation matrix. We store only the first two unit axes of the rotation matrix and calculate the third axis as the cross product of others.

Learned DMPs can be reproduced with new conditions. The first step of reproduction is the observation of the initial state, that is, the virtual frame approximated using the measured fingertip positions. We parametrize the DMPs using the initial state, the desired goal state and the desired action time. Then, we integrate the dynamical systems and obtain a concrete object trajectory. In order to execute this trajectory, we transform the whole trajectory to each of fingertip frames, assuming that the contacts are intact during the motion, i.e., no slip or roll occurs. This implies that the initial transforms between the object frame and the contact frames do not change. Finally, these trajectories are converted to joint-space using inverse kinematics.

Reproduced joint-space trajectories are then used to generate the manipulation signal u_m using PID control. We activate the virtual springs prior to the execution of a trajectory. We assume that a stable and manipulable grasp configuration is given. Virtual springs generate the grasp signal u_g to maintain the object in contact. Manipulation and grasping signals are combined to obtain the control signal $u_c = u_m + u_g$.

In the following subsections, we detail the underlying methods: The VSF and the DMP. Then, we specify the implementation details.

A. Virtual Spring Framework

We define virtual springs between fingers and the virtual object frame as shown in Fig. 1.a. These springs generate the grasping forces to hold the manipulated object. We also use force feedback at the fingertips in order to adjust the stiffness of springs during manipulation.

We organize the components of VSF into 3 subsections. First, we define the virtual object frame which is used in the subsequent sections. Then, we present the rules for calculating grasping force generated by the springs. Finally, adaptive stiffness control using force feedback is described.

1) Virtual Object Frame: We follow the previous approaches [9] and [16] in calculation of the virtual object pose. We only modify the orientation formula in order to account for more than 3 fingers. Approximate object position p_o and orientation R_o are calculated using the fingertip positions p_i:

$$\mathbf{p}_o = \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i \tag{1}$$

$$\begin{aligned}
\mathbf{R}_o &= [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z] \in SO(3)\\
\mathbf{r}_x &= \frac{\mathbf{p}_n - \mathbf{p}_1}{\|\mathbf{p}_n - \mathbf{p}_1\|}\\
\mathbf{r}_z &= \frac{(\mathbf{p}_m - \mathbf{p}_1) \times \mathbf{r}_x}{\|(\mathbf{p}_m - \mathbf{p}_1) \times \mathbf{r}_x\|}\\
\mathbf{r}_y &= \mathbf{r}_z \times \mathbf{r}_x
\end{aligned}$$

We assume for n fingers, $\mathbf{p}_n$ is the position of the thumb. Positions of all fingers other than the index $\mathbf{p}_1$ and the thumb are merged into $\mathbf{p}_m = \sum_{i=2}^{n-1}\mathbf{p}_i/(n-2)$.

2) Grasp Force: The manipulated object is kept in grip by the forces generated by the virtual springs. The following equation is used to calculate grasp force for finger i:

$$\mathbf{f}_{gi} = K_{gi}(\|\Delta\mathbf{p}_i\| - L_i)\frac{\Delta\mathbf{p}_i}{\|\Delta\mathbf{p}_i\|} \tag{2}$$

$\Delta\mathbf{p}_i = \mathbf{p}_o - \mathbf{p}_i$ is the vector between positions of i^{th} fingertip and the virtual frame. L_i is the rest length of the spring connecting i^{th} fingertip to the virtual frame origin.

We follow a simple strategy to determine rest lengths for holding the object in grasp. The springs are activated when the object is in contact with the fingertips. At the same moment, L_i is set as a proportion of the actual distance between the virtual frame and the fingertip frame $l_k\,\|\Delta\mathbf{p}_i\|$. We use $l_k = 0.9$ in this paper since lower values distort the grasp shape by dragging outer fingers closer to the center and higher values result in weaker grasps.

The stiffness $K_{gi}, i = 1, 2...n$ are given a common value initially and adapted using the force feedback while the object is in grasp. Stiffness adjustment is explained in the following subsection.

We add these forces to the trajectory control command for combining force and positions controls. In order to do this, we calculate joint torques $u_g = \{u_{gi}, i = 1, 2...n\}$ to achieve desired Cartesian forces. The forces f_{gi} are transformed to joint-space torques using manipulator Jacobians J_i:

$$u_{gi} = J_i^{\mathsf{T}} f_{gi} \tag{3}$$

3) Adaptive stiffness: Stiffness of a spring is updated using the force data that is read from the corresponding sensor. The robot is expected to have sensors that monitor the forces applied on fingertips. Magnitudes of these forces are controlled indirectly by adjusting the spring stiffness.

Since the movement of fingers is subject to disturbances, finger contacts may break or apply too much force to disturb the object pose. Adapting stiffness increases contact stability during the manipulation.

Stiffness controller is a simple proportional controller with fingertip force magnitude as the setpoint, current force magnitude reading as the process variable and stiffness of the corresponding spring as the control variable. Stiffness control requires a proportional control gain K_k. In our experiments,

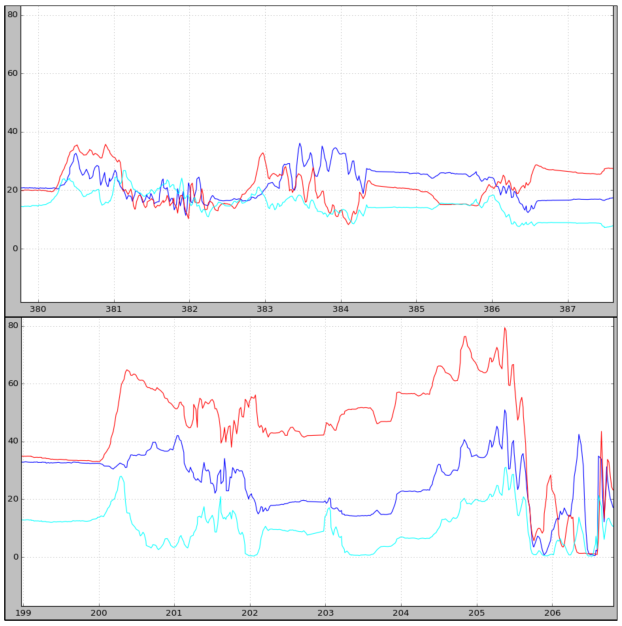


Fig. 2. Measured force magnitudes at three fingertips during the translation action with (top image) and without adaptive stiffness control (bottom image). The object is lost during the manipulation without adaptive stiffness. The parameters are $K_{gi} = 20, K_k = 0.001$ for the adaptive, $K_{gi} = 40$ for the static cases. The thumb force is excluded as it has a larger scale.

a $K_k = 0.001$ value is used. We determined this gain by experimentation as explained in Section IV-B.1. Higher values of K_k may lead to unstable grasps while lower values may fail to response disturbances effectively.

Please note that this is a soft gain that allows a reasonable steady-state error. It gives fingertip forces some independence to deviate from the target value in order to achieve manipulation. Figure 2 shows the force magnitudes of fingers during manipulation. It can be seen that there is a considerable margin around the target value of 20.

B. DMP Formulation

DMP formulation consists of a set of differential equations which describe the evolution of a dynamical system. Commonly, a point-attractor equation is used as the *transformation system* to ensure convergence to a goal state [4]. A forcing term f is added to the system in order to shape the trajectory of y. A phase variable x defines a temporal relationship among the basis functions.

$$\tau\ddot{y} = \alpha_z(\beta_z(g - y) - \tau\dot{y}) + f \tag{4}$$

$$f = \frac{\sum_{i=1}^{N} \psi_i w_i}{\sum_{i=1}^{N} \psi_i} x \tag{5}$$

$$\tau\dot{x} = -\alpha_x x \tag{6}$$

In this one dimensional example, state variable y is accelerated towards the goal g as a damped spring model. α_z and β_z are gain constants. Timing of the whole system can be scaled with the addition of a new parameter τ. The forcing term f consists of Gaussian basis functions ψ_i. w_i denote the weight of i^{th} basis function. The forcing term can be shaped by modifying the weights which are usually learned from demonstration. Learning is usually done using function approximators [4].

This formulation of a DMP is just one out of many in the literature. There exist different notations and modifications. In this work, we use the Kulvicious' DMP formulation [7] that allows smoothly joining multiple primitives and also inherits previous enhancements to preserve the shape of the trajectory after scaling and rotating. We omit the primitive joining property in this particular work. [7] modifies the phase variable x and the Gaussian basis ψ_i as follows:

$$\dot{x} = -\alpha_x \frac{exp(\frac{\alpha_x}{\Delta t}(\tau T - t))}{(1 + exp(\frac{\alpha_x}{\Delta t}(\tau T - t)))^2} \tag{7}$$

$$\psi_i = exp(-(\frac{t}{\tau T} - c_i)^2)/2\sigma_i^2 \tag{8}$$

This formulation uses x as only a weighting variable and handles the phasing using time variables: current time t, total time T and the sampling time Δt. Modifications of ψ_i place basis functions evenly along the time. Here, c_i is the center and σ_i is the variance of the i^{th} kernel.

The DMP formulation is designed to preserve the shape of the learned motion when the initial state y (at time 0), the final state g or the time scaler τ are modified. Generalization capabilities of DMPs are discussed in detail in [4] and [7].

In our work, a DMP is learned for each of the object frame position and orientation dimensions. Hence, the DMPs represent the dynamic behavior of the object frame. Integration of the DMP during run-time produces the reference object frame H_t which was introduced earlier (Fig. 1.a).

C. Implementation Details

Our robot system consists of an Allegro hand mounted on a UR5 6-DOF arm. Allegro hand has 4 fingers with 4 independently controllable joints each. Fingers of the robot are equipped with Optoforce OMD 20-SE-40N sensors that can measure 3-DOF force at their tips with 1 *kHz* frequency.

We implemented our method as *robot operating system (ROS)* programs. The UR5 arm is controlled using *MoveIt!* library[1]. The Allegro hand inverse kinematics and manipulator Jacobian are calculated using *Orocos KDL* library[2]. We use *DMPBBO* implementation of the DMP and function approximators [15]. We use Kulvicious' DMP formulation [7] and locally weighted regression function estimator provided in this library. The codebase of our experiments is available online[3].

IV. EXPERIMENTS

The experiments are designed to evaluate the generalization capabilities of our method. Ideally, our method should preserve the properties of the DMP and the virtual spring

[1]moveit.ros.org
[2]www.orocos.org
[3]github.com/gokhansolak/lfd-experiments-iros2019

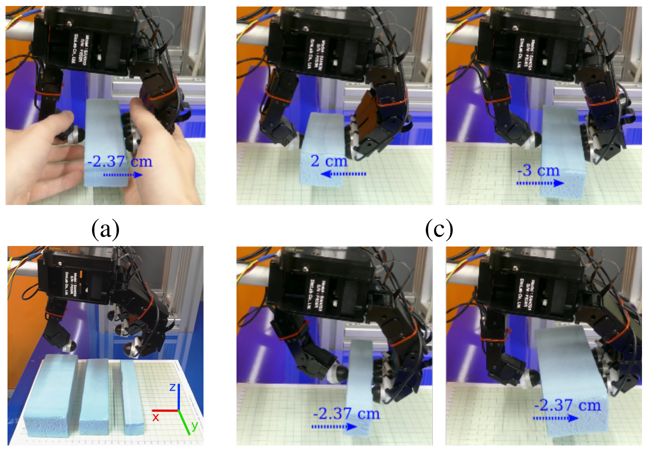


Fig. 3. (a) Kinesthetic demonstration of the translation action. (b) Allegro hand, task reference frame and 3 objects used in the experiments. (c) Reproduction with different initial and goal positions. (d) Reproduction with different object sizes. Blue arrows show desired translations.

frameworks. The DMP framework allows reproduction of learned trajectories with different initial and goal states. Virtual springs allow the robot to keep hold of objects of unknown shape during manipulation which enables reproduction of the learned motions on objects with different sizes.

We also provide a comparison of static and adaptive stiffness approaches for virtual springs. This comparison is necessary to evaluate the effect of force feedback. The translation action is used as the basis of comparison.

The next subsection contains the details of our experiment setup. Then, we present the results of force feedback and generalization experiments.

A. Experiment setup

The UR5 arm is only used to bring the hand in reach of the object and pull it off the table after grasping. When the hand is brought to the reach of the object, a human places the fingertips of the robot on grasping points by intuition. Then, the virtual springs are activated. The object is pulled off the table, a hand manipulation action is executed and the object is put back on the table. This procedure is common for all experiments. In the experiments that use force feedback, it is activated together with the springs. Force feedback is inactive during kinesthetic teaching since the external forces may introduce instability.

We carry all experiments for translation and rotation actions that are learned once from demonstration. The translation action moves the object by -2.37 *cm* in *x*- axis. The rotation action rotates the object by roll-pitch-yaw angles 55°,17°,56°. The reference coordinate frame is shown in Fig. 3.b. The initial and final states of manipulation actions are shown in Fig. 1.b.

Our experiments contain 3 rectangular prism objects of different sizes as shown in Fig.3.b. The small one has $25 \times 150 \times 25$, the medium one has $44 \times 150 \times 34$ and the large one has $60 \times 150 \times 45$ *mm* dimensions. All objects are held by their parallel surfaces lying along the *yz*- axes (Fig. 3.b). We use the medium sized object for learning the translation and rotation actions.

TABLE I
COMPARISON OF ADAPTIVE AND STATIC STIFFNESS APPROACHES

		Stable	Intact	e_p (cm)	e_R (rad)
Static stiffness	20	3	1	0.525	0.542
	40	1	0	0.239	0.1267
	60	0	0	-	-
Adaptive stiffness gain	0.0001	3	3	1.139	0.168
	0.0005	3	3	0.464	0.270
	0.001	3	3	**0.360**	**0.087**
	0.005	0	0	-	-

TABLE II
GENERALIZATION EXPERIMENTS

		Total Transform		Average Final Error	
Motion	Change	x_p (cm)	x_R (rad)	e_p	e_R
Translate	Original	-2.37	0.18	0.59	0.10
	Goal to -1	-1.0	0	0.39	0.14
	Initial to -2 Goal to 0	2.0	0	0.57	0.26
	Initial to 1 Goal to -2	-3.0	0	0.41	0.16
	Small object	-2.37	0.18	0.47	0.29
	Large object	-2.37	0.18	0.86	0.35
Rotate	Original	1.63	1.40	0.66	0.63

B. Results

1) Contribution of Force Feedback: We tested 3 cases of static stiffness ($K_g = 20, 40, 60$) and 4 cases of stiffness adaptation gains ($K_k = 0.0001, 0.0005, 0.001, 0.005$) with the initial stiffness of 20. We repeated each case 3 times on the translation action. The results are presented in Table I. *Stable* is the count of trials that finished without dropping the object, *intact* is the count of trials that maintained all finger contacts. The average final state errors for position e_p and orientation e_R are also given. The results show that the force feedback improves the stability of the grasp and decreases the final state error. We use the adaptive stiffness with a gain of 0.001 in the rest of the experiments as it is stable and yields the minimum average error.

2) Generalization Experiments: We apply 3 types of generalization; object size, initial state and goal state. The translation action is repeated for multiple task conditions. The rotation action is tested as a proof of concept. Each case is repeated 5 times with slight changes in contact positions. Changes in contact positions are arbitrary results of human-provided grasps.

We reproduce the learned motion with initial and goal states which are different than that of the demonstrated motion. In our problem specification, a state corresponds to the pose of the object in the hand frame. For initial and goal state modification, we test 3 different conditions. Considering that the original translation is from 0 to -2.37 *cm* on the *x*-axis, modified conditions are as follows: Scaling down the translation by moving the goal to -1 *cm* on the *x*- axis, reversing the direction by moving the initial state to -2 *cm* and the goal to 0 *cm* (Fig. 3.c, left image), and shifting the grasp pose by moving the initial state to 1 *cm* and the goal to -2 *cm* (Fig. 3.c, right image).

Granted that a plausible grasp is given for an object, we generalize the learned actions for similar objects. We apply the translation action that is learned on the medium object to the small and the large objects as shown in Figure 3.d. No parameter tuning is done for DMPs, VSF nor force feedback.

Table II presents the average final state errors of 5 repetitions of each task. We present the position error e_p in *cm* and the orientation error e_R in *radians*. These error values are calculated by taking the absolute difference between the desired final pose and the actual final pose of the virtual frame. We also present the total transform expected to be generated by an action both in position x_p and orientation x_R as a reference.

As it is seen on Table II, it is possible to generalize a learned hand manipulation action to new conditions. The error values may increase slightly in some cases, but the change is insignificant. Our method can transfer the generalization capabilities of DMPs to dexterous manipulation problem with the virtual springs and force feedback enhancements.

V. CONCLUSION

We can learn and generalize dexterous hand manipulation skills by a combination of DMPs, the VSF and force feedback. Our method answers the trajectory execution and maintaining stable grasp problems simultaneously.

We show that our method can achieve dexterous manipulation of objects without having access to the object model. However, the control algorithm would benefit from more information about the object. We used force sensors to increase the stability of a grasp, however, these sensors require the assumption that a contact point is limited to the fingertip. More information about contact can be retrieved using distributed tactile sensors [17]. A possible future work is to use such sensors to infer useful information about an object such as its weight, its center of mass and the surface normals at the contact points. This information can be used to regulate the fingertip forces in order to increase the contact stability and generalize to new objects with various weights and mass distributions.

We show that the VSF helps generalization of object size on the objects with a prism shape. However, its robustness with irregularly shaped objects needs further validation.

Kinesthetic teaching is a practical solution to transfer human skills to the robot. However, this system may benefit from different teaching methods such as data gloves or visual tracking devices in order to learn more complex manipulation trajectories.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] Z. Chen, C. Ott, and N. Y. Lii. A compliant multi-finger grasp approach control strategy based on the virtual spring framework. In *International Conference on Intelligent Robotics and Applications*, pages 381–395. Springer, 2015.

[3] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Trans. Robotics*, 30(4):816–830, 2014.

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[5] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[6] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[7] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, 28(1):145–157, 2012.

[8] M. Li, Y. Bekiroglu, D. Kragic, and A. Billard. Learning of grasp adaptation through experience and tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3339–3346. Ieee, 2014.

[9] M. Li, H. Yin, K. Tahara, and A. Billard. Learning object-level impedance control for robust grasping and dexterous manipulation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6784–6791. IEEE, 2014.

[10] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–340, Nov 2011.

[11] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[12] M. Regoli, U. Pattacini, G. Metta, and L. Natale. Hierarchical grasp controller using tactile feedback. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 387–394, 2016.

[13] A. Shukla and A. Billard. Coupled dynamical system based hand-arm grasp planning under real-time perturbations. *Robotics: Science and Systems VII*, page 313, 2012.

[14] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell. Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 464–470. IEEE, 2015.

[15] F. Stulp. Dmpbbo–a c++ library for black-box optimization of dynamical movement primitives, 2014.

[16] K. Tahara, S. Arimoto, and M. Yoshida. Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4322–4327. IEEE, 2010.

[17] T. P. Tomo, A. Schmitz, W. K. Wong, H. Kristanto, S. Somlor, J. Hwang, L. Jamone, and S. Sugano. Covering a robot fingertip with uskin: A soft electronic skin with distributed 3-axis force sensitive elements for robot hands. *IEEE Robotics and Automation Letters*, 3(1):124–131, 2017.

[18] J. Umlauft, D. Sieber, and S. Hirche. Dynamic movement primitives for cooperative manipulation and synchronized motions. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 766–771. IEEE, 2014.

[19] K. Van Wyk and J. Falco. Calibration and analysis of tactile sensors as slip detectors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2744–2751, 2018.

[20] F. Veiga, J. Peters, and T. Hermans. Grip stabilization of novel objects using slip prediction. *IEEE Transactions on Haptics*, 11(4):531–542, 2018.

[21] T. Wimböck, C. Ott, A. Albu-Schäffer, and G. Hirzinger. Comparison of object-level grasp controllers for dynamic dexterous manipulation. *The International Journal of Robotics Research*, 31(1):3–23, 2012.