Using dVRK Teleoperation to Facilitate Deep Learning of Automation Tasks for an Industrial Robot

Jacky Liang[1], Jeffrey Mahler[1], Michael Laskey[1], Pusong Li[1], Ken Goldberg[1,2]

***Abstract*— Deep Learning from Demonstrations (Deep LfD) is a promising approach for robots to perform precise bilateral automation tasks involving contact and deformation, where dynamics are difficult to model explicitly. Deep LfD methods typically use datasets of 1) human videos, which do not match robot kinematics and capabilities or 2) waypoints collected with tedious move-and-record interfaces, such as teaching pendants or kinesthetic teaching. We explore an alternative using the Intuitive Surgical da Vinci, which combines a pair of gravity-balanced, high-precision, passive, and 6-DOF master arms with stereo vision, allowing humans to teleoperate precise surgical automation tasks. We present DY-Teleop, an interface between the da Vinci master manipulators and an ABB YuMi industrial robot to facilitate the collection of time-synchronized images and robot states for deep learning of automation tasks involving deformation and dynamic contact. The system has an average latency of 194ms and executes commands at 6Hz. We present YuMiPy, an open source library and ROS package for controlling an ABB YuMi over Ethernet. Data collection experiments with scooping a ball into a cup, untying a knot in a rope, and pipetting liquid between two containers suggest that demonstrations obtained by DY-Teleop are comparable with those by kinesthetic teaching in demonstration time. We performed Deep LfD for the scooping task and found that the policy trained with DY-Teleop achieved a $1.8\times$ higher success rate than a policy trained with kinesthetic teaching. Code, videos, and data are available at `berkeleyautomation.github.io/teleop`.**

I. Introduction

In Deep Learning from Demonstrations (LfD), a human demonstrates a task and a robot learns a policy from the demonstrations using deep learning [1]. LfD has been used to for robot assembly tasks [2], autonomous driving [3], and deformable object manipulation [4], [5].

While one approach to train a policy is reinforcement learning, this can in practice require many robot trials [6]. Deep LfD [7] is a promising approach to LfD for tasks with dynamics and state spaces that are difficult to model explicitly, such as deformable object manipulation. In Deep LfD, the policy is a deep neural network that maps high dimensional states such as color images directly to robot motion commands. Due to high dimensionality and long time horizons, Deep LfD methods typically require many demonstrations [7].

Popular methods for data collection include (1) kinesthetic teaching, where an operator manually moves a robot's arms to complete a task [8], [9], [10], and (2) low-DOF teleoperation, where an operator controls a subset of a robot's end-effector pose or joints through a device such as a phantom omni [11] or video game controller [7]. However, these interfaces have several shortcomings. First, providing demonstrations using these interfaces can be tedious because the user may be restricted to controlling a subset of the available DOFs at a time. It may be difficult to control two arms simultaneously [12] or have fine control over gripper closing force. Second, these interfaces may introduce artifacts into the data that make learning difficult. For example, during kinesthetic teaching, recorded images may contain occlusions from the hands of the demonstrator and motor torques may contain spikes from human interaction with the arms. Master-slave bilateral teleoperation systems such as those used in robot-assisted surgery [13] or based on virtual reality [14] can mitigate these issues by enabling users to directly control a robot's end-effector through master arms designed to facilitate natural human motions.

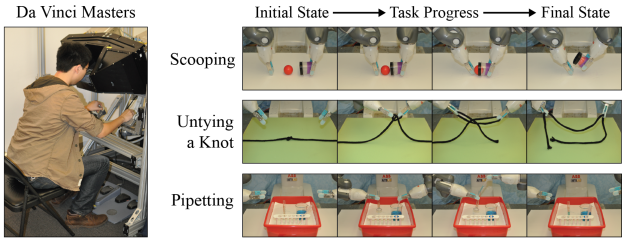

Fig. 1: DY-Teleop, our bilateral teleoperation system. (Left) The operator controls the 6-DOF pose of each arm and gripper width through an Intuitive Surgical da Vinci bilateral Masters manipulator. (Right) The robot executes the operator's commands to collect demonstrations of 3 tasks: scooping a ball into a cup, untying a knot, and pipetting liquid from a beaker to a graduated cylinder.

In this paper we explore the use of a da Vinci master manipulator surgical teleoperation system [15] to collect demonstrations for three bilateral manipulation tasks with the ABB YuMi, an industrial robot, and we study the use of Deep LfD to perform one of the tasks. Our system, dVRK-YuMi-Teleoperation (DY-Teleop), enables real-time and simultaneous control of the end-effector poses and closing forces of both parallel-jaw grippers, and it is illustrated in Figure 1. The operator has access to stereo video feeds of the robot's workspace and controls the robot through the arms of the master manipulator. The operator's controls are mapped to the reference frame of the robot and executed on the YuMi. To facilitate controlling the YuMi we develop YuMiPy, an open source Python library for controlling the YuMi over Ethernet.

The AUTOLAB at UC Berkeley (`automation.berkeley.edu`)
[1] Department of Electrical Engineering and Computer Sciences; {jackyliang, jmahler, mdlaskey, alanpusongli}@berkeley.edu
[2] Department of Industrial Engineering and Operations Research; goldberg@berkeley.edu
[1−2] University of California, Berkeley; Berkeley, CA 94720, USA

We study data collection performance on three tasks involving dynamic contact and deformable object manipulation: scooping a ball into a cup, untying a knot, and pipetting liquid from a beaker to a graduated cylinder. Our data collection experiments suggest that DY-Teleop can collect demonstrations at a comparable speed to kinesthetic teaching and is 21% faster for rope untying, which involves significant use of both arms simultaneously. We also train a Deep LfD policy that maps images to end-effector pose commands for the scooping task using demonstrations collected with our system, and the policy has a $1.8\times$ higher success rate than one trained with kinesthetic teaching.

II. Related Work

Programming robots to perform manipulation tasks that involve dynamic motions and trajectories or interactions with the environment is difficult, and Learning from Demonstrations (LfD) methods have the potential to address these challenges [1]. The two stages of LfD are collecting human demonstrations and deriving a robot policy from demonstration data. Methods used for collecting human demonstrations can be divided into two groups: teleoperation and kinesthetic teaching [16], [17], [18].

A. Teleoperation

In teleoperation, a user operates an input device, or "master," to control the robot, or "slave," which executes the controls specified by the master. Early teleoperation systems such as Mosher's Handyman used hydraulic arms for manipulation in unsafe environments, such as near radioactive materials or in space [19]. Due to the high cost and complexity of early systems, recent research in LfD has considered low-DOF input devices such as the Phantom Omni [11] and the Xbox controller [7], which can limit the ability of the operator to simultaneously control multiple degrees of freedom at once [1], and devices that mount to human arms [20], [19], such as the Intuitive Surgical da Vinci master manipulators [5]. However, direct teleoperation of the end-effector pose has the challenge of redundant degrees of freedom as there can be multiple inverse kinematic solutions for a certain pose [21]. This has motivated the use of motion tracking systems, which can be markerless [22], marker-based [23], [24], or a full body suit (e.g. XSens MOVEN [25]) that can map human joints to robot joints. Teleoperation has been used for demonstrating motions for pouring water [20], grasping objects [26], [23], and suturing in robot assisted surgery [5]. Teleoperation systems induce latency through communication and control delays, bandwidth constraints, and limited feedback (e.g. vision only) which can hinder the operator's ability to perceive the workspace and apply accurate controls [19]. Recent research in teleoperation has explored predicting an operator's future actions using inverse optimal controls [27] and from visual features [28], as well as teleoperating an exoskeleton robot via brain-machine interfaces [29].

B. Kinesthetic Teaching

In kinesthetic teaching, an operator physically moves the robot in a gravity-balanced "teach mode" to complete the given task [8], [9], [4], [10] and the robot records the motion. Kinesthetic teaching has been used to give demonstrations for stacking towers [30], scooping melons [31], and writing on a whiteboard [32] among other tasks, and has recently become popular in LfD due to its intuitive interface for directly controlling joint angles and promising experimental results. Lee and Ott [25] used online kinesthetic teaching to provide corrective motions to incrementally refine desired trajectories. Wrede et al. [21] proposed a method for using kinesthetic demonstrations to learn more desired joint configurations for certain end-effector poses that have multiple inverse kinematics solutions. Kinesthetic teaching is challenging for tasks where the operator must move multiple arms at once, and it can introduce artifacts into the state space (e.g. hands visible in the camera, which are undesirable for learning a policy from images).

C. Comparison of Teleoperation and Kinesthetic Teaching

Akgun et al. [33], [11] compared user preferences for teleoperation versus kinesthetic teaching in giving demonstrations for unilateral manipulation tasks such as scooping and pouring coffee beans and stacking blocks using a PR2. The study found kinesthetic teaching is easier to use than teleoperating PR2's end-effectors using a Phantom Omni haptic device. The authors also argued that a non-expert may learn how to give kinesthetic demonstrations faster than through a secondary interface, which is needed for teleoperation. In another study [34], participants were asked to give trajectory demonstrations of pushing and closing a box via an HOAP-3 robot. The study found that users who were dissatisfied with kinesthetic teaching were concerned about not knowing the robot's technical details, e.g. the field of view of the vision system, and how the robot will use trajectories to learn. Another user study by Muxfeldt et al. [35] on industrial assembly discovered that a human's performance on a task did not translate into relative performance difference when giving kinesthetic demonstrations. Our work compares teleoperation with kinesthetic teaching in giving demonstrations for Deep LfD.

D. Learning from Demonstrations

Our work is also closely related to research on robotic learning from human demonstrations. Abbeel et al. [36] learned a policy for helicopter acrobatics from expert demonstrations given with a joystick controller. Calinon and Billard [37] studied how to extract implicit joint and workspace constraints from kinesthetic demonstrations, and Phillips et al. [38] used kinesthetic teaching to learn door opening motions on a PR2. Schulman et al. [39] used Thin Plate Splines to warp kinesthetically demonstrated trajectories of rope tying with a PR2 to solve novel test cases. Ross et al. [40] learned a policy for collision avoidance from monocular images with a UAV by iteratively demonstrating corrective feedback actions. Laskey et al. [41] extended

this work to only collect corrective actions for novel states classified with a One-Class SVM. The authors later learned policies for singulating (or separating) objects from a pile using supervised learning on data collected by a hierarchy of human supervisors [4] that provided demonstrations through a joystick and a GUI through Amazon Mechanical Turk [7]. We use supervised learning following the approach of Laskey et al. [7] to train policies collected with our bilateral teleoperation system.

III. Teleoperation System

The goals of DY-Teleop are (a) to enable an operator to quickly provide successful demonstrations of precise bilateral robotic tasks and (b) to collect synchronized data of trajectories of the robot's end-effector poses, gripper widths, and images of the workspace for deep learning of robot control policies.

A. System Architecture

Figure 2 illustrates the DY-Teleop architecture. During teleoperation the most current 3D poses of the two manipulators, along with the two current gripper widths, are mapped from the Masters Tool Manipulators (MTMs) to the YuMi arms. Using the most current MTMs' poses, a pose mapping service computes the corresponding end-effector poses for the YuMi robot. This pose forwarding pipeline is asynchronous to prevent mismatch in pose command frequency, as the YuMi may take longer than the operator to execute a motion. Finally, the machine that interfaces with the YuMi robot constantly pulls the most current desired poses and gripper widths to command the YuMi accordingly. If the clutch foot pedal is held by the user, then teleoperation pauses, allowing the user to move the MTMs without moving the YuMi. This is useful for executing motions extending outside of the masters' workspace.

B. Masters-Slave Pose Mapping

A pose mapping service translates the current MTM poses to the desired YuMi poses. Our system processes relative pose changes of the operator into relative poses for the YuMi because the initial poses of the MTMs and YuMi robot arms may change. To describe the pose mapping equations we define the following frames and poses:

- m - World frame of the MTMs. Poses published by the masters are with respect to this frame.
- w - World frame of the YuMi. Poses commanded to the YuMi are with respect to this frame.
- mi - Frame of initial MTM pose. ${}^{m}T_{mi}$ is the first pose of an MTM.
- mc - Frame of the current MTM pose. ${}^{m}T_{mc}$ is the current pose of an MTM.
- yi - Frame of the initial YuMi pose. ${}^{w}T_{yi}$ is the first pose of an YuMi arm.
- yc - Frame of the current YuMi pose. ${}^{w}T_{yc}$ is the current pose of an YuMi arm.

Formally, we denote a pose from frame a to frame b as ${}^{b}T_{a} = (\mathbf{R}, \mathbf{t}) \in SE(3)$ where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are respectively the rotation and translation from frame a to b. Note ${}^{b}T_{a} = {}^{a}T_{b}^{-1}$.

At every time step we use the current MTM pose ${}^{m}T_{mc}$ to compute the desired current YuMi arm pose ${}^{w}T_{yc}$. The following derivations are done for one MTM and YuMi arm pair. In practice, pose-mapping is performed for both arms, and the pose transformation equations are identical.

We define ${}^{m}T_{w}$ to be a pure rotational transform that rotates a given pose to a frame used by the YuMi arm to a frame used by an MTM. While it is feasible to compute ${}^{w}T_{yc}$ by applying the relative transform of ${}^{m}T_{mc}$ to ${}^{m}T_{mi}$ onto ${}^{w}T_{yi}$ as follows:

$$ {}^{mi}T_{mc} = {}^{mi}T_{m}\,{}^{m}T_{mc} \tag{III.1} $$

$$ {}^{w}T_{yc} = {}^{w}T_{m}\,{}^{mi}T_{mc}\,{}^{m}T_{w}\,{}^{w}T_{yi}\,, \tag{III.2} $$

this often produces unintuitive results for the operator. For example, if the operator moves an MTM to the right, the above mapping would move the corresponding YuMi arm also to the right provided that the MTM was not rotated prior to moving to the right. If the operator rotated the MTM in place, then moving to the right for the operator will no longer move the YuMi arm to the right. Thus we define a reference frame that the operator uses to perform demonstrations, and the mapping of ${}^{w}T_{mc}$ to ${}^{w}T_{yc}$ uses the relative transformations of the MTMs and YuMi arms in this reference frame.

Denote the operator reference frame as r, then the transformation that takes any ${}^{m}T_{mc}$ to ${}^{r}T_{mc}$ is ${}^{r}T_{m}$, which is a pure rotational transformation where its rotation is equal to that of ${}^{m}T_{mi}$. Then the new pose mapping computation is:

$$ {}^{mi}T_{mc} = {}^{mi}T_{m}\,{}^{m}T_{mc} \tag{III.3} $$

$$ {}^{r}T_{mc} = {}^{r}T_{m}\,{}^{mi}T_{mc}\,{}^{m}T_{r} \tag{III.4} $$

$$ {}^{w}T_{yc} = {}^{w}T_{m}\,{}^{r}T_{mc}\,{}^{m}T_{w}\,{}^{w}T_{yi}\,. \tag{III.5} $$

To expand a human operator's reachable workspace we use the clutch foot pedal. When the pedal is pressed down, the motions of YuMi arms are paused, allowing the operator to move the MTMs to a new pose. When the pedal is released, teleoperation resumes and the YuMi arms are responsive to the MTMs again. To accomplish this, we note that the relative transformation from ${}^{mi}T_{mc}$ to ${}^{m}T_{mi}$ after clutch up should not consider the transformation that occurred from clutch down to clutch up. Furthermore, before any clutching has occurred, we can treat ${}^{m}T_{mi}$ as the pose at the last clutch up because current MTM transformations are computed relative to it.

We define the following frames and poses related to clutching:

- r - Reference frame of the operator.
- u - Frame of MTM at last clutch up. ${}^{m}T_{u}$ is the MTM pose at clutch up.
- d - Frame of MTM at last clutch down. ${}^{m}T_{d}$ is the MTM pose at clutch down.

${}^{m}T_{u}$ is initialized to ${}^{m}T_{mi}$ and ${}^{mi}T_{u}$ is initialized to the identity transformation. At every clutch up we perform the

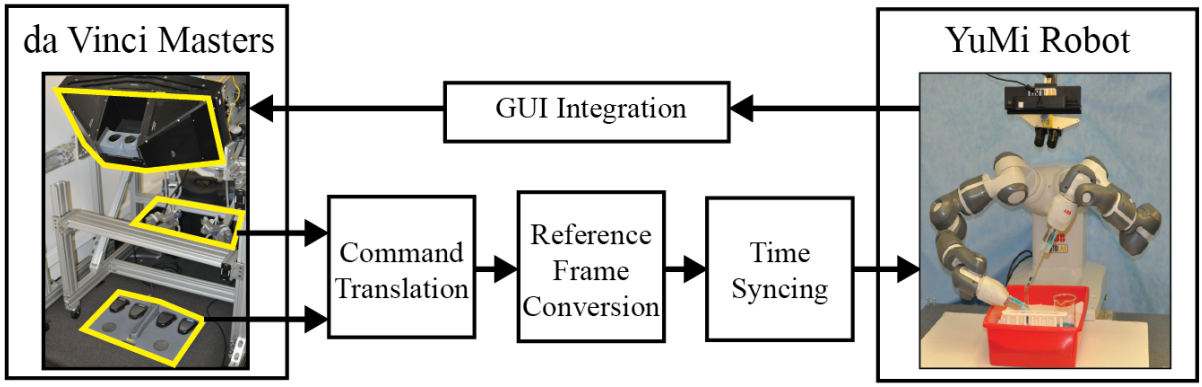

Fig. 2: System architecture for our real-time teleoperation software. (Bottom-Left to Bottom-Right) We first translate user inputs from the da Vinci Masters Tool Manipulators (MTMs) into end-effector poses and gripper widths are discretized. We then convert the poses of the MTMs into the reference frame of the YuMi, handling operations such as clutching to expand the operator's reachable workspace. Finally the pose and gripper width commands are simultaneously executed on the YuMi. (Top-Right to Top-Left) Images from a stereo camera from the robot are integrated into a Graphical User Interface that displays a set of options and state information to the user on the Masters vision interface.

following update:

$$ {}^{mi}T_u \leftarrow {}^{mi}T_u \, {}^{u}T_m \, {}^{m}T_d \tag{III.6} $$

$$ {}^{m}T_u \leftarrow {}^{m}T_{mc} \, . \tag{III.7} $$

Incorporating clutch to the pose mapping computation we arrive at the following equations, which are calculated at every time step, that receives the current MTM pose ${}^{m}T_{mc}$ and outputs the desired YuMi arm pose ${}^{w}T_{yc}$:

$$ {}^{mi}T_{mc} = {}^{mi}T_u \, {}^{u}T_m \, {}^{m}T_{mc} \tag{III.8} $$

$$ {}^{r}T_{mc} = {}^{r}T_m \, {}^{mi}T_{mc} \, {}^{m}T_r \tag{III.9} $$

$$ {}^{w}T_{yc} = {}^{w}T_m \, {}^{r}T_{mc} \, {}^{m}T_w \, {}^{w}T_{yi} \, . \tag{III.10} $$

C. Hardware Overview

1) ABB YuMi: The YuMi robot is a human-safe, stationary industrial robot with two 7-DOF arms. Each arm has 0.02mm positional accuracy[1] and a payload of 250g. The parallel-jaw grippers have a maximum closing force of 20N and a maximum gripper width of 5cm. The original gripper tips were replaced with custom silicone ones [42]. The YuMi supports kinesthetic teaching through motion controllers that provide gravity-compensation.

2) da Vinci Master Manipulators: The dVRK masters has two 6-DOF MTMs that can output their current 3D poses, and each MTM has a gripper-control interface that can output variable widths. The dVRK masters also has a set of foot pedals that provide high-level user interactions, such as starting, ending, and pausing a teleoperation session, as well as toggling between different camera views. The user can simultaneously control the MTMs, the gripper widths, and the foot-pedals. DY-Teleop does not have haptic feedback.

3) Cameras: To provide stereoscopic vision, we mount two Logitech C270 webcams at 1.2m above the YuMi robot work space, centered around the robot. This height was chosen such that the field of view of the two cameras can include a large area of the workspace. The separation between the two cameras can be adjusted, but it is generally set around 6.4cm, the average pupillary distance for U.S. adult males [43]. Another Logitech C270 webcam is placed at level with the workspace to provide an additional perspective that gives greater precision to manipulating objects near the work surface. Video feeds of resolution 640×480 from these cameras are relayed via separate USB 2.0 connections to be displayed as left and right eyed videos on the viewing screens of the dVRK masters system.

D. Software Overview

1) YuMi Python (YuMiPy) Interface: The YuMi is designed to be programmed by ABB's RAPID language, which requires uploading programs to the robot that cannot be changed dynamically. To increase programming flexibility, we developed YuMiPy[2], a custom python library for interfacing with the ABB YuMi robot over Ethernet.

YuMiPy consists of a Python client running on a user desktop and a RAPID TCP/IP server running on the robot controller that is based on the OpenABB library[3]. The API supports position commands, gripper functionality, and changes to parameters such as movement speed. The API also provides functions to stream the current end-effector poses, joint angles, and motor torques of both arms from the YuMi robot, allowing for data collection of the YuMi robot during execution. YuMiPy also supports commanding the robot through a ROS service.

To command the robot, a user first creates a Python client that establishes an Ethernet connection to the server. The client maintains separate processes for asynchronous socket communication with the YuMi to prevent blocking the user's main code while commands are being executed. Python calls to the YuMiPy API send requests to the asynchronous communication processes via a queue, and each process iteratively reads the most recent command and forms a packet. Each packet consists of a numeric code for the command, data corresponding to the command (e.g. joint angles), and an error checking code. Packets are sent to the server, which parses the packet, executes the corresponding

[1] new.abb.com/products/robotics/yumi/

[2] berkeleyautomation.github.io/yumipy

[3] github.com/robotics/open_abb

RAPID command, and sends an acknowledgement or error message back to the Python client. YuMiPy has a mean latency of 3.89ms for one way communication between the Python client and the RAPID server on our system.

2) Network Communication: DY-Teleop uses several libraries to synchronize poses and images over the network. We use the Python multiprocessing library to send data between processes on the same machine using thread-safe queues. Data is communicated over the network using ROS services, publishers, and subscribers in ROS Jade.

IV. Deep LfD

In Deep Learning from Demonstrations, a human provides demonstrations to a robot, and a policy mapping observed states to controls is learned. In this work states are images of the workspace.

We can formalize this as follows: denote a policy as a measurable function $\pi : \mathcal{X} \rightarrow \mathcal{U}$ from images $\mathcal{X}$ to control inputs $\mathcal{U}$, such as position commands to a robot. We consider policies $\pi_\theta : \mathcal{X} \rightarrow \mathcal{U}$ parameterized by some $\theta \in \Theta$, such as the weights of a neural network. Under our assumptions, any such policy π_θ induces a probability density over the set of trajectories of length T:

$$p(\tau|\pi_\theta) = p(\mathbf{x}_0) \prod_{t=0}^{T-1} p(\mathbf{x}_{t+1}|\pi_\theta(\mathbf{x}_t), \mathbf{x}_t)\,,$$

where a trajectory τ of length T is a sequence of state and action tuples: $\tau = \{(\mathbf{x}_t, \mathbf{u}_t)\}_{t=0}^{T-1}, \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)\,.$

We note this assumes the Markov property in the state space. We collect demonstrations using DY-Teleop and kinesthetic teaching from a supervisor's policy π_{θ^*}, where θ^* may not be contained in Θ. We assume the supervisor is not necessarily optimal, but achieves a desired level of performance on the task.

We measure the difference between controls using a surrogate loss $l : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}$ [44], [45]. The surrogate loss we consider is the squared euclidean loss on the control $l(\mathbf{u}_1, \mathbf{u}_2) = ||\mathbf{u}_1 - \mathbf{u}_2||_2^2$. The objective of LfD is to minimize the expected surrogate loss under the distribution induced by the robot's policy:

$$\min_\theta \; E_{p(\tau|\pi_\theta)} \sum_{t=0}^{T-1} ||\pi_\theta(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)||_2^2\,. \tag{IV.1}$$

However, in practice this objective is difficult to optimize because of the coupling between the loss and the robot's distribution on states. Thus, we instead minimize an upper-bound on this objective [7] via sampling N trajectories from the supervisor's policy.

$$\min_\theta \sum_{n=0}^{N-1} \sum_{t=0}^{T-1} ||\pi_\theta(\mathbf{x}_{t,n}) - \pi_{\theta^*}(\mathbf{x}_{t,n})||_2^2\,; \quad \tau \sim p(\tau|\pi_{\theta^*})\,. \tag{IV.2}$$

V. Experiments

We collected demonstration data for three tasks illustrated in Figure 1 and evaluated the performance of Deep LfD on the scooping task. All experiments were run on a system comprised of: three desktops running Ubuntu, the ABB YuMi, two Logitech C270 webcams, the da Vinci masters, two TP-Link Gigabit Ethernet switches, and a CN-160 light on the workspace to help the operator perceive depth.

A. System Latency

Our system was able to execute commands on the YuMi at 6Hz. The total latency between a user movement and the visual feedback of the completed motion was 194ms. This consisted of 7ms to convert pose commands, 155ms to transmit and execute commands on the YuMi using YuMiPy, and 31ms to transmit images from USB to the da Vinci master manipulator viewer.

B. Tasks

Figure 1 illustrates the following tasks.

1) Scooping: The goal of the scooping task is to scoop a ball into a cup with the free gripper and the lip of the cup, which is challenging to model due to dynamic rolling and collisions.

Initially the robot grasps a cup with the left gripper and holds the right gripper in a paddle position. The ball is placed in a set of predefined points on a grid between the grippers. During demonstration, the robot, by a combination of pushing via the right gripper and scooping with the left gripper, scoops the ball into the cup. A demonstration is successful when the ball rests inside the cup and is only supported by the cup. A common failure is the ball bounces and rolls away after it collides with the gripper or cup.

2) Untying a Knot: The goal of the rope untying task is to untie an overhand knot. This is difficult because the task involves deformation, has multiple steps, and requires precise gripper placements to pull only a desired segment of the rope.

Initially a rope tied in an overhand knot is placed on a foamed surface before the robot. As it is hard for the rope to slip along the surface of the grippers' tips, the operator cannot untie the rope as a human normally would. Instead, the robot unties the knot by iteratively regrasping and pulling the end of the rope loose. The task is complete when the rope no longer makes a loop with itself. The most common failure is accidentally grasping multiple segments of the rope when trying to pull a single segment of the rope loose.

3) Pipetting: The goal of the pipetting task is to transfer 3mL of fluid from a beaker to a 10mL graduated cylinder using a plastic pipette. The task is difficult due to deformations of the plastic pipette, requirement of fine control of the gripper width, the multi-step nature of the task, and the precision needed to put the pipette tip into the graduated cylinder with only a 15mm diameter opening.

Initially the beaker, graduated cylinder, and pipettes are set at fixed locations in the workspace. During demonstration, the operator grasps the graduated cylinder with the right

Task	Mode	Time to Completion (s) Median	Mean	Min	Max	Std	Trials
Scooping	K	**0.91**	0.96	0.30	2.62	0.33	50
	T	2.49	2.64	1.53	8.62	1.13	50
Rope Untying	K	268.74	267.92	138.80	396.23	105.10	5
	T	**213.16**	241.80	144.81	362.50	76.91	5
Pipetting	K	**106.03**	110.42	85.14	143.46	23.93	6
	T	187.38	200.31	152.05	267.17	36.43	6

TABLE I: Comparison of demonstrations collected using kinesthetic teaching (K) and DY-Teleop (T). We compare the time it takes to collect demonstration data for each task and interface. DY-Teleop is faster to collect successful demonstrations for the rope untying tasks.

gripper and the pipette with the left gripper. The pipette is then brought to the beaker and the operator opens the gripper to a desired width to draw fluid. Next, the pipette is transported to the opening of the graduated cylinder and the operator closes the gripper to push the liquid out of the pipette. This action is repeated until 3mL are transferred into the graduated cylinder. Common failure modes include dropping the pipette, failing to extract liquid from the beaker, and colliding with the transparent objects in the workspace.

C. Analysis of Demonstrations

A demonstration is successful if the operator reaches the goal state. For each task a pre-defined strategy for both demonstration modes was determined, and strategies for all three tasks are described above. Demonstrations that deviated from them were not included in our analysis.

We collected 50 demonstrations of the scooping task, 5 of the knot untying task, and 6 of the pipetting task. The scooping demonstrations were collected by a different operator than the pipetting and rope untying. The difference in demonstration numbers was due to the time required to demonstrate each task. Each demonstration was started and stopped by the operator using a set of footpedals to select options from a GUI. For all demonstrations we recorded the time of completion and synchronized streams of poses, joints, and motor torques for both of YuMi's arms as well as RGB images from the overhead webcams sampled at 10Hz.

1) Comparison with Kinesthetic Teaching: To evaluate our system, we compared the DY-Teleop demonstrations with kinesthetic ones. During kinesthetic teaching, the operator moves the YuMi arms through direct physical contact. To control the gripper width and grasp force, the operator uses the FlexPendant, a handheld operator unit that provides user inputs to the YuMi robot.

Table I summarizes timing results. The median time to collect demonstrations using DY-Teleop was comparable to kinesthetic teaching and is 21% faster for the rope untying task, which involves significant use of both arms simultaneously. Kinesthetic teaching was faster for the scooping task because two fast and short wrist movements while holding the end-effectors can complete the task, whereas for teleoperation, slight delays in the system made it more difficult for the operator to confidently execute fast motions. Kinesthetic was also faster for pipetting because it is difficult for the operator in teleoperation to discern the location of the small opening of the 5mL graduated cylinder.

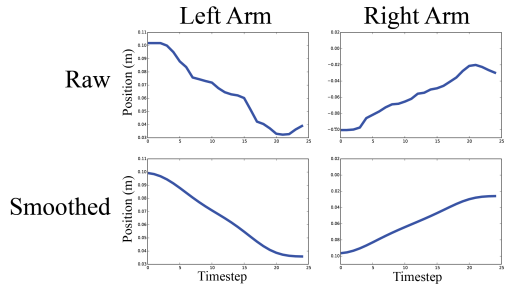

Fig. 3: Comparison of raw and filtered end-effector pose trajectories for the scooping task. The raw data contains high-frequency components as the grippers move toward the ball in the center of the table. We use a Gaussian filter to smooth the trajectories, which can facilitate learning on small datasets.

2) Preprocessing: Raw actions and observations collected from DY-Teleop and kinesthetic teaching are not suitable for machine learning due to high-frequency changes in end-effector pose and excess data collected at the beginning and end of each trajectory due to time mismatch between the demonstration start and stop.

Removing noise from human demonstrations has been shown to improve Deep LfD performance [46], so to smooth the trajectories we applied a Gaussian filter to the pose trajectories with a standard deviation of 3 time steps for translation and 2 time steps for the Euler rotation angles. Figure 3 shows an example trajectory before and after smoothing.

To remove excess background from the images we cropped each image to a window around the grippers. To be invariant to lighting changes we binarized each channel of the color images.

D. Scooping with Deep LfD

To compare Deep LfD performance trained with demonstrations from DY-Teleop with those with kinesthetic teaching, we train a deep policy on images from the scooping task. The state space for this task is an overhead RGB image with dimensions $100 \times 250 \times 3$. Similar to [4], we apply a binary mask to the image to be robust to changes in lighting conditions. The control space is $\mathcal{U} = \{\Delta x_l, \Delta y_l, \Delta \theta_l, \Delta x_r, \Delta y_r, \Delta \theta_r\}$, where x and y are the planar positions of the grippers in the frame of the YuMi and θ is the gripper wrist angle that is varied during the scooping task.

The neural network architecture used is the same as in [4] and is implemented in Tensorflow. It has a convolutional layer and two fully connected layers, all with Rectified Linear Unit (ReLu) activation. The network is trained on a GeForce 980 GPU using stochastic gradients with momentum. During training, we scale each dimension in the control vector to the range $[-1, 1]$.

We compare the neural net policy against 3 methods for physical evaluation. The first is an open-loop policy that performs a fixed scooping motion in the middle of the workspace. The second is a trained policy with 50 demonstrations collected via kinesthetic teaching with the

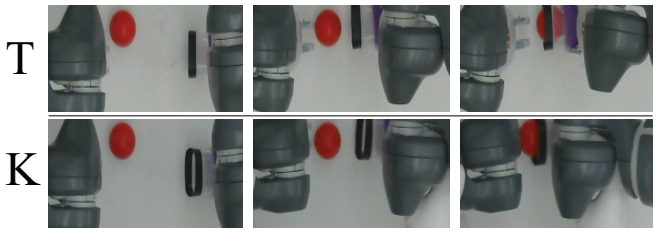

Fig. 4: Two trajectories collected from DY-Teleop (T) and kinesthetic teaching (K). The primary difference between the images are the gloves visible in the bottom-right of the image for the kinesthetic method.

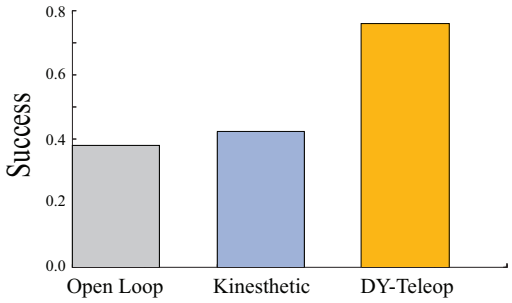

Fig. 5: The success rates for the trained scooping policies over 21 placements. The three methods are an open-loop demonstration that is not learned, a policy trained with kinesthetic demonstrations, and a policy trained with the DY-Teleop . The policy trained with DY-Teleop has a $1.8\times$ higher success rate than one trained with kinesthetic teaching. We attribute this to the hands being in the state space during training with kinesthetic teaching.

supervisor wearing white gloves to match the color of the background. The third is a trained policy with 50 demonstrations collected via DY-Teleop . During training the ball was placed uniformly in a 4×8cm grid.

Results are illustrated in Figure 5. We evaluate the policies on 21 ball placements in the 4×8 cm grid. The policy trained with demonstrations collected from the DY-Teleop has a $1.8\times$ higher success rate than one trained with demonstrations from kinesthetic teaching. We attribute this discrepancy to the mismatch in test and training distributions that occurs when the supervisor places their hands in the state space while collecting demonstrations via kinesthetic teaching. Despite the supervisor wearing gloves with similar features to the background, deep networks may be prone to over-fitting to subtle feature changes [47].

VI. Discussion and Future Work

We present DY-Teleop, a system that interfaces between the Master Tool Manipulators of an Intuitive Surgical da Vinci and an ABB YuMi that enables bilateral teleoperation for demonstrating automation tasks involving dynamic contact and deformation.

While we successfully learned a deep policy for bilateral scooping, we did not learn a policy for the rope untying and pipetting tasks because they require multiple steps and failure recovery actions that are known to make learning difficult without segmenting the task [15]. Figure 6 illustrates how the demonstrations may be segmented. In future work we plan to improve system reliability and reduce latency, as well as explore hierarchical machine learning to recover task structures automatically.

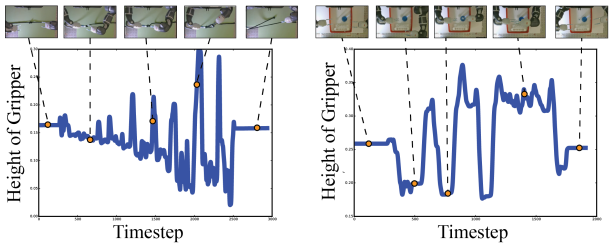

Fig. 6: Illustration of the hierarchical structure in the demonstrations collected with DY-Teleop for the knot untying (left) and pipetting (right) tasks. The knot untying demonstrations consist of a set of repeated grab-and-pull actions as the knot was initially loosened which gradually had longer pulls until the rope came undone. The pipetting demonstrations consist of several segments where the gripper raises the pipette out of a container, transports it along a linear path, and sets it down in a different container.

Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the Real-Time Intelligent Secure Execution (RISE) Lab, and the CITRIS "People and Robots" (CPAR) Initiative. The authors were supported in part by the U.S. National Science Foundation under NRI Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems, the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program, the Berkeley Deep Drive (BDD) Program, and by donations from Siemens, Google, Cisco, Autodesk, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Sponsors. We thank our colleagues who provided helpful feedback and suggestions, in particular Anca Dragan, Sanjay Krishnan, Ruta Joshi, Zoe McCarthy, and Sammy Staszak.

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] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 827–834.

[5] A. Murali, A. Garg, S. Krishnan, F. T. Pokorny, P. Abbeel, T. Darrell, and K. Goldberg, "Tsc-dl: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4150–4157.

[6] S. M. Kakade *et al.*, "On the sample complexity of reinforcement learning," Ph.D. dissertation, 2003.

[7] M. Laskey, C. Chuck, J. Lee, J. Mahler, S. Krishnan, K. Jamieson, A. Dragan, and K. Goldberg, "Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[8] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss, "Learning manipulation actions from a few demonstrations," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1268–1275.

[9] K. Kronander, M. Khansari, and A. Billard, "Incremental motion learning with locally modulated dynamical systems," *Robotics and Autonomous Systems*, vol. 70, pp. 52–62, 2015.

[10] 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.

[11] B. Akgun, K. Subramanian, and A. L. Thomaz, "Novel interaction strategies for learning from teleoperation." in *AAAI Fall Symposium: Robots Learning Interactively from Human Teachers*, vol. 12, 2012, p. 07.

[12] C. Chen, S. Krishnan, M. Laskey, R. Fox, and . Ken Goldberg1, "An algorithm and user study for teaching bilateral manipulation via iterated best response demonstrations," in *Automation Science and Engineering (CASE), 2017 IEEE International Conference on*. IEEE, 2017.

[13] A. M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Industrial Robot: An International Journal*, vol. 31, no. 6, pp. 499–508, 2004.

[14] T. Takahashi and H. Ogata, "Robotic assembly operation based on task-level teaching in virtual reality," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 1083–1088.

[15] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," in *International Symposium of Robotics Research. Springer STAR*, 2015.

[16] P. G. De Barros and R. W. Linderman, "A survey of user interfaces for robot teleoperation," 2009.

[17] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[18] S. Lichiardopol, "A survey on teleoperation," *Dept. Mech. Eng., Dynamics Control Group, Technische Universiteit Eindhoven, Eindhoven, Dept., Mech. Eng., Dyn. Control Group, The Netherlands, Tech. Rep. DCT2007*, vol. 155, 2007.

[19] T. B. Sheridan, *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[20] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 763–768.

[21] S. Wrede, C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil, "A user study on kinesthetic teaching of redundant robots in task and configuration space," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 56–81, 2013.

[22] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 183–202, 2008.

[23] L. Lukic, J. Santos-Victor, and A. Billard, "Learning robotic eye–arm–hand coordination from human demonstration: a coupled dynamical systems approach," *Biological cybernetics*, vol. 108, no. 2, pp. 223–248, 2014.

[24] Y. Wu, Y. Su, and Y. Demiris, "A morphable template framework for robot learning by demonstration: Integrating one-shot and incremental learning approaches," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1517–1530, 2014.

[25] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2-3, pp. 115–131, 2011.

[26] K. Hsiao and T. Lozano-Perez, "Imitation learning of whole-body grasps," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5657–5662.

[27] C. Schults, S. Gaurav, L. Zhang, and B. Ziebart, "Goal-predictive robotic teleoperation from noisy sensors," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[28] T. M. Y. L. Akio Namiki, Yosuke Matsumoto, "Vision-based predictive assist control on master-slave systems," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[29] S. Qiu, Z. Li, W. He, L. Zhang, C. Yang, and C.-Y. Su, "Brain–machine interface and visual compressive sensing-based teleoperation control of an exoskeleton robot," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 1, pp. 58–69, 2017.

[30] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 2012, pp. 391–398.

[31] A.-L. Pais Ureche and A. Billard, "Learning bimanual coordinated tasks from human demonstrations," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*. ACM, 2015, pp. 141–142.

[32] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3970–3975.

[33] B. Akgun and K. Subramanian, "Robot learning from demonstration: kinesthetic teaching vs. teleoperation," *Unpublished manuscript*, 2011.

[34] A. Weiss, J. Igelsböck, S. Calinon, A. Billard, and M. Tscheligi, "Teaching a humanoid: a user study with hoap-3 on learning by demonstration," in *Proceedings of 18th IEEE International Conference on Robot and Human Interface Communication*, 2009.

[35] A. Muxfeldt, J.-H. Kluth, and D. Kubus, "Kinesthetic teaching in assembly operations–a user study," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 533–544.

[36] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[37] S. Calinon and A. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Advanced Robotics*, vol. 23, no. 15, pp. 2059–2076, 2009.

[38] M. Phillips, V. Hwang, S. Chitta, and M. Likhachev, "Learning to plan for constrained manipulation from demonstrations," *Autonomous Robots*, vol. 40, no. 1, pp. 109–124, 2016.

[39] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Robotics Research*. Springer, 2016, pp. 339–354.

[40] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[41] M. Laskey, S. Staszak, W. Y.-S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 462–469.

[42] M. Guo, D. V. Gealy, J. Liang, J. Mahler, A. Goncalves, S. McKinley, and K. Goldberg, "Design of parallel-jaw gripper tip surfaces for robust grasping," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[43] N. A. Dodgson, "Variation and extrema of human interpupillary distance," in *Electronic imaging 2004*. International Society for Optics and Photonics, 2004, pp. 36–46.

[44] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning."

[45] S. Ross and D. Bagnell, "Efficient reductions for imitation learning."

[46] C. Chuck, M. Laskey, S. Krishnan, R. Joshi, and K. Goldberg, "Statistical data cleaning for deep learning of automation tasks from demonstrations," in *Automation Science and Engineering (CASE), 2017 IEEE International Conference on*. IEEE, 2017.

[47] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.