Teaching Robots Using Mixed Reality

Samir Yitzhak Gadre

Brown University, Department of Computer Science

samir_yitzhak_gadre@brown.edu

April 19, 2018

Advisor & First Reader: Prof. George Konidaris
Second Reader: Prof. Stefanie Tellex

Contents

Abstract

Learning from Demonstration (LfD) is an emerging field in robotics wherein a robot learns a skill given demonstrations from a teacher. We present a novel method for providing expert training data for robot manipulators. We use a Mixed Reality (MR) head mounted device (HMD) that presents a holographic sphere on the end-effector. The sphere can be manipulated to move the robot arm. Our system allows the user to specify the importance of specific moments in their demonstration and segment a demonstration into multiple skills. It also allows the user to visualize and manipulate various parameters between training and testing time to adapt to environment changes. We demonstrate the efficacy of our system by training a robot to perform pick and place tasks autonomously.

1. Introduction

As robots become integrated into everyday life, it will be necessary to develop channels for humans to teach machines with actions and words rather than with programming languages. Learning from Demonstration (LfD) fills this void, in that it allows people to teach robots with their own actions. While there is room for algorithmic improvement within the field of LfD, training interfaces are a major bottleneck.

Existing training methods are often not scalable. Current approaches involve moving joints by hand or via 2D graphical user interfaces (GUIs). However, physically manipulating a robot is not always practical as the robot can be large, heavy, or its operating environment dangerous. For example, robots are useful in high radiation environments where humans cannot travel safely. If someone wanted to teach a robot a new skill in such an environment, it would be better to do so virtually. 2D GUIs are in wide use but require an external monitor, split attention between robot and screen, and are awkward to use.

Traditional methods also put strain on teachers to iterate on their teaching style and provide completely new demonstrations to clear up ambiguity. This is particularly a problem if the teacher is unskilled. Currently, it is also hard to visualize what a robot has learned prior to actual execution. So, it is difficult for a teacher to detect and prevent failure.

There is a unique opportunity to use Mixed Reality (MR)—also known as augmented reality—to address these limitations. As MR devices such as the Microsoft HoloLens become prevalent and affordable, they turn into viable interfaces for human and robot interaction. The current and future states of the robot can be shown by a hologram. Furthermore, a user can interact with the hologram to change the state of the real robot. In this way, a robot operator can get realistic representation of the robot without being in physical proximity to the robot and without requiring the robot to actually execute a motion. Because MR augments the world view, a user is still able to pay attention to the real world. Since it is possible to represent the robot directly, the interface does not

pose an additional barrier between the teacher and the robot. Finally, holograms can be used by the robot to convey its level of understanding before it executes a task in the real world. We describe an approach that uses MR to collect training data for LfD.

2. Background

We are concerned with the problem of allowing humans to train robots without having to worry about low-level implementation details. Our approach is based on a procedure called Learning from Demonstration (LfD).

2.1. Learning from Demonstration

In LfD, we model the world as a Markov Decision Process (MDP). The goal is to learn a policy π, as defined in (1), which describes how to act in the MDP to maximize a reward function (minimize a loss function). We adapt our notation from that of Argall et. al. [1]:

$$\pi : S \rightarrow A. \tag{1}$$

Here $S \subseteq \mathbb{R}^n$ is the set of states that the agent can take, where each state is represented by an n dimensional vector. $A \subseteq \mathbb{R}^m$ is the set of actions the robot can take, where each action is represented by an m dimensional vector. Transitions between states are modeled by a probabilistic transition function T:

$$T(s'|s,a) : S \times A \times S \rightarrow [0,1]. \tag{2}$$

T gives the probability of being in current state $s' \in S$ given the previous state $s \in S$ and the action $a \in A$ taken at s. There is a set of demonstration examples E for a skill, where $e_i \in E$ represents the i-th example. Each example trajectory e_i then is the sequence shown in (3) with k observations:

$$e_i = [(s_1^i, a_1^i), (s_2^i, a_2^i), \ldots, (s_k^i, a_k^i)]. \tag{3}$$

Here $s_j^i \in S$ is the j-th observed state and $a_j^i \in A$ is the j-th observed action in example trajectory e_i. By training on E, the goal is to reproduce the policy that generated the example trajectories:

$$\pi^* : S \rightarrow A. \tag{4}$$

Given this framework, there are many experimental and algorithmic choices to be made—for example the representation of observations and the action spaces, which determine the demonstration space. One might use all reachable coordinates $(x, y, z) \in \mathbb{R}^3$ to represent the observation space and the corresponding displacement vectors (dx, dy, dz) between any two observations to define the set of actions.

Whereas the above formulation lets us decide what information to collect, the teacher and robot relationship determines how the demonstrations will be collected. One common approach is to use kinesthetic teaching where a teacher manipulates a robot while it

records observations about its state [2]. This kind of demonstration has been used to learn striking motions for table tennis [3]. Other popular approaches use teleoperation and mimicking. In the former, a human demonstrates a motion by moving the robot via a controller. In the latter, a rig might be used so a human's motion is captured for robotic imitation. A mapping between human and robot joints is then used [2]. Mimicking has been show to be effective method to teach hand and arm movements [4, 5, 6]. Once E is collected, supervised learning is often used to learn π^* [7].

2.2. Dynamic Movement Primitives

DMPs are a type of policy often used with LfD. They were first formulated as a control mechanism for decoupled degrees of freedom (DOFs) and take inspiration from the mechanics of a damped spring [7]. The following is the formulation of DMPs presented by Pastor et al. [8]:

$$\tau \dot{v} = P(g - x) - Dv + (g - x_0)f, \tag{5}$$

$$\tau \dot{x} = v, \tag{6}$$

where x_0 is the start, x is the current position, g is the goal, v is the velocity, τ, D, and P are scaling factors, and f is a non-linear forcing function. The system is loosely non-linear. Without the f term, (1) represent the linear dynamics of a spring-damper. The linear portion of (1) is essentially a PD controller governed by parameters P and D, where P is the proportional gain and D scales the damping term. In practice, it is important that D is set such that the spring analog is critically damped to preventing oscillations [2]. The linear portion of (1) ensures that we steadily make progress from x to g. The function f takes the following form and determines the shape of the curve we take from x_0 to g:

$$f(\phi) = \frac{\sum_{i=1}^{n} \psi_i(\phi) w_i \phi}{\sum_{i=1}^{n} \psi_i(\phi)}. \tag{7}$$

ϕ is a phase variable, each ψ_i is a basis function such as the Radial or Fourier basis [2][8]. Each w_i is a weight that can be learned using supervised regression methods on a set of training examples E [9]. The number n of basis functions captures the trade-off between being able to represent more complex functions and over-fitting on E as n becomes larger.

2.3. User Interfaces

Our work is particularly influenced by the findings of Akgun et. al. [10] that discuss the usefulness of keyframes in a LfD framework. Here keyframes are taken to be a sparse set of data points, such that moving from point to point will complete the desired task. This work on keyframes inspired us to create Critical Points (CPs)[1], which are states in the demonstration that are critical to performing a skill. We also draw inspiration from work of Elliott et. al. [11], which uses 2D GUIs to allow for "adaptation" after training,

[1]CPs are discussed in more detail in section 3.3 and 3.4.

where a teacher is able to tweak a demonstration offline to create more robust motion plans. The concept of "adaptation" influences our MR endpoint visualization approach presented in section 3.5.

User studies by Rosen et. al. show that MR is an effective platform for robots to communicate future movements to human trainers. Time-lapse animation of a robot arm's motion in MR were displayed so a human observer could understand the intent of the robot [15]. Furthermore the use of MR has obvious safety benefits, both for the robot and for the teacher. By training and simulating on a hologram, it is possible to isolate failure modes.

Other methods explore MR for motion planning in industrial robots. MR has been used to allow users to mark obstacles, starting, and goal positions within a state space [12]. MR via-points points generated by users have been used for Bayesian network based path fitting algorithms [13, 14]. There have also been methods to visualize trajectories, where there are options for user feedback and retraining [14].

3. Approach

We describe key elements of our approach—calibration, representation, demonstration, data processing, and execution—to address the task of allowing a teacher to train a manipulator to autonomously perform pick and place objectives. Calibration takes place by moving a hologram of the robot onto the real robot. The representation section presents our choices of state and action spaces. The demonstration section discusses how the MR system is used to collect a demonstration. The data processing section addresses the preprocessing on a demonstration before a policy is created. Finally, the execution section explains how a user gets the robot to generalize demonstrated motion trajectories.

3.1. Calibration

We rely on two main coordinate systems, the Unity[2] coordinate system U and the ROS coordinate system R. Both have position (x, y, z), orientation (r, p, y), and equivalent scales. U and R are global frames for the HoloLens application and Baxter respectively. The origin of U is determine by the position and orientation of the HoloLens at the start of the application. The origin of R is determined by the base link of the Baxter robot, which is located at the center of the robot's torso. Hence there is a critical calibration step in which the user must, implicitly, determine the transforms between these two frames. We solve this problem by having a holographic rendering of the robot, dubbed the shadow, appear with its base link at the origin of U. The shadow is initialized such that it has all of the same joint angles as the real robot.

The user is tasked with calibrating the system by moving the shadow until it is flush with the real robot, thereby generating the a transform Tf and Tf^{-1}:

$$Tf : U \rightarrow R, \tag{8}$$

[2]Unity is 3D game engine used to develop applications for HoloLens.

$$(Tf)^{-1} : R \rightarrow U. \tag{9}$$

3.2. Representation

The state of the robot arm is specified by the position of the Baxter end-effector in R concatenated with two booleans g and cp. It is easy to find the location of the end-effector in the R by applying successively transforms that start at the base link and move through the Baxter arm. g takes value 1 if the gripper is open and 0 otherwise. cp takes value 1 if the user determines the position is a CP in the demonstration and 0 otherwise. Here a CP is used as a key-point or support for the demonstration, signifying that the particular state achieved is crucial to the integrity of the demonstrated skill.

Actions are displacement vectors that take the end effector from starting state s to next state s' over a timestep. The timestep is determine by the frame rate of our Unity application as well as networking speed that enables connection between the holographic application and the Baxter base station. A sequence of observations of states and actions over multiple timesteps constitutes a demonstration.

3.3. Demonstration

Once calibrated, a transparent sphere appears at the location of the of the end-effector as in **Figure 1**. The sphere acts as a controller for the end-effector. As a user moves the sphere in MR, the real gripper follows. As the arm moves, the shadow is updated. Tf and Tf^{-1} are used to deal with change of coordinate systems. Baxter is put into zero-gravity mode to ensure that the ending location of the sphere is the end location of the end-effector. Voice commands "open" and "close" can be used to toggle the gripper.

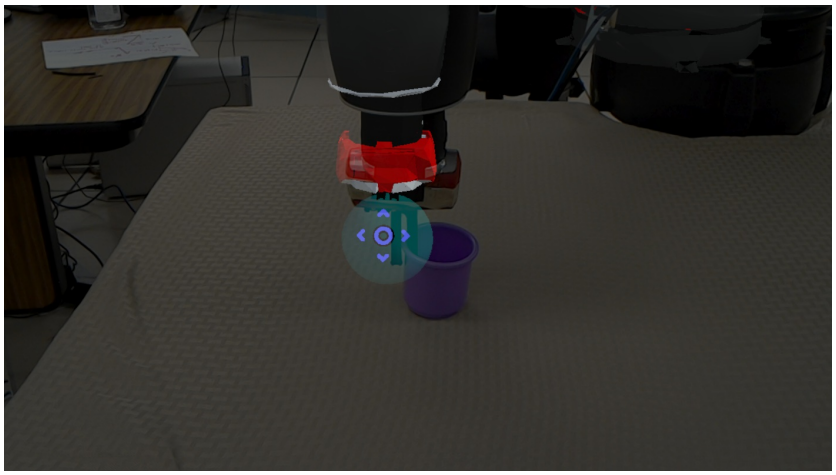
Our teleoperation system can be used to collect demonstrations for learning. Using the key word "start" enables recording of the state over time. The (x, y, z) location in R is recorded after the robot has been moved and the corresponding gripper state is saved. We assume that the user is unskilled, and hence the each recorded position has default cp value 0. This signifies that the specific position is not important, but rather a point's relative position in the demonstration.

However, by clicking on the sphere during the demonstration, the user specifies that the particular position is important. To represent this the cp value associated with the position is set to 1. The notion of CPs becomes useful for demonstrating a fine motor skill in which exact placement is important. Alternatively, it is a helpful notion for someone who is unskilled and provides noisy demonstration. Being able to specify CPs allows the system to smooth out motion trajectories.

When the user clicks on the sphere a snapshot of the shadow is saved at the given time step as in **Figure 2**. The rendering of this shadow stays in the user's holographic vision to provide both a visual reminder of the CP and a marker that shows where the arm was in the past.

To make our system more robust to chained demonstrations, where a user wants to execute many skills that they deem different, we train several DMPs for various pieces of

(a) *The start position with a superimposed transparent control sphere.*



(b) *The arm is teleoperated downwards to clamp the cup.*

(c) *The user continues to manipulate the sphere to pick up the cup.*

Figure 1: *An example of teleoperation to pick up a cup using our system.*

(a) *The start state of the arm, where the sphere is initially clicked.*

(b) *An Intermediate CP point after start. Notice that the position during the first CP is saved for visualization.*

(c) *A third CP is saved, in this case, at the stop location.*

Figure 2: *Three saved CPs over time.*

the demonstration. For example, **Figure 3** shows a 2D example of what different segments of a demonstration might look like. The user has the ability to start a new segment of a demonstration by using the voice command "new". When the user opens or closes the gripper during demonstration, it is automatically assumed that the system should start start an entry to store data for a new DMP. Each segment then has its own data points. The user says the word "stop" to indicate that they have finished their demonstration.

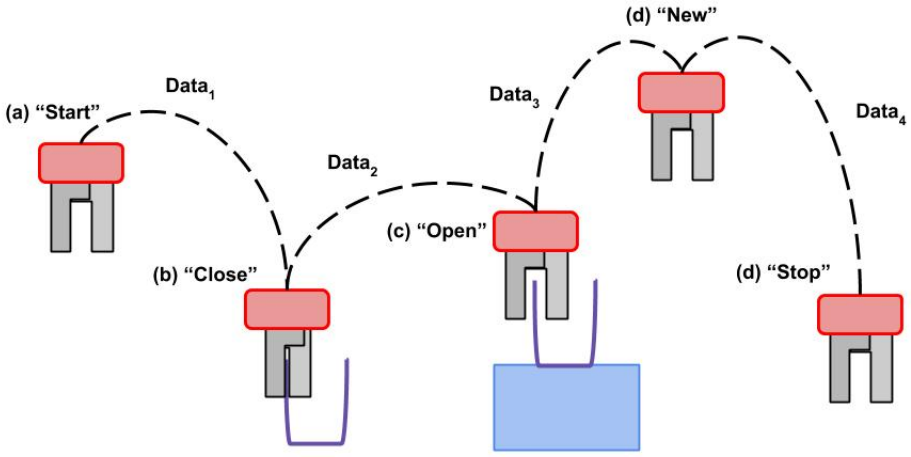


Figure 3: *This figure shows the movement of a gripper over time and voice commands in quotation. A user starts a new segment of a demonstration using keywords. $Data_i$ represents observations between successive letters in the diagram. This data is used to train the i-th DMP. Keywords are uttered at each letter to begin a new segment of the demonstration. We hence have endpoints labeled* ***(a)–(e)****.*

3.4. Data Processing

Instead of learning a DMP for each segment based on the raw data, we opt to do the following based on stored CPs. We start by parameterizing data points in each dimension (x, y, z) by time step for a given segment of the demonstration. Without loss of generality, we discuss the x dimension on the j-th segment. We get a function $x(t)$ that is defined at integral points $[1, ..., n]$ where $t = 1$ is the first time step and $t = n$ is the n-th time step. We assume that the user provided us with with a noisy demonstration and only the positions that are specified as CPs need to be achieved. We define the residual in the standard way:

$$r_x(i) = x(i) - f_x(i), \tag{10}$$

where $r_x(i)$ is the i-th residual in the x dimension, i is the i-th time step, and f_x is the function we are trying to approximate. The degree of f_x is the minimum of (1) the number of CPs $+ 1$, (2) 10, and (3) n. We then use a weighted least square objective function to

account for CPs. Our objective is to minimize the following error function err_x:

$$err_x = \left(\sum_{i=1}^{n} \frac{r_x(i)}{k_i} \right)^2. \tag{11}$$

Here k_i takes value 0.01 if the user deemed the i-th position a CP and 1 otherwise. Hence error associated with CPs will be weighted more heavily and our minimization of err_x will naturally result in a curve that is close to important parts of the demonstration. **Figure 4** shows an example of biased curve fitting to smooth over data for one DMP in all three dimension. Three CPs have been chosen.

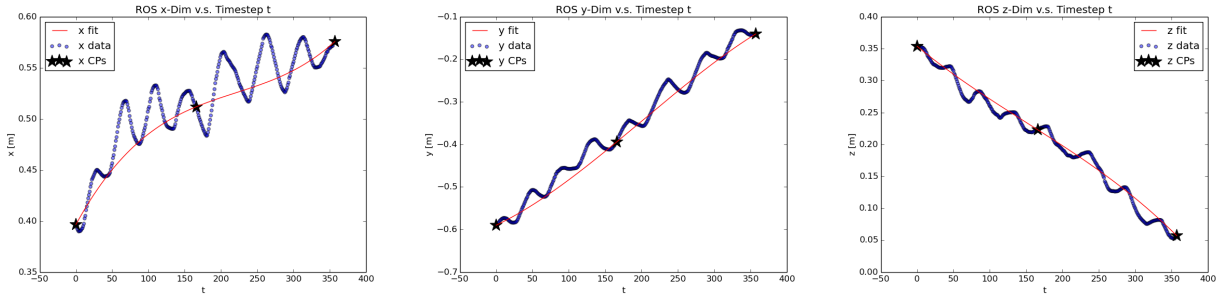


Figure 4: *Noisy data from the demonstration in **Figure 2**. However, CPs support the demonstration and allow us to conduct path smoothing.*

Once we have found f_x, f_y, and f_z. We pass our sequence of filtered data points $(f_x(i), f_y(i), f_z(i))$ for $i = 1 : n$ into a DMP solver which gives us a model for the segment, parameterized by a start and end state [16].

3.5. Execution

After training, smaller spheres appear to the user, as shown in **Figure 5**. These spheres take the position of the endpoints of segments of the demonstration. Between each successive pair of spheres there is a policy defined by the DMP. It is now possible for the user to perturb these spheres, perhaps to adapt to change in the real world. For example, if a user has trained the robot to pickup a cup, perhaps the location of the cup has changed. The system works in such a way that, in the general case, the end of one segment is the beginning of the next. The relative order of the actions is preserved. To execute the motion plan, the user says the command "execute", and the robot moves linearly to the start position to executes the DMP motion plans and gripper actions. The plans take roughly the same amount of time to execute as the demonstration itself.

4. System

Source code for our implementations can be found on the holocontrol_baxter branch of holobot, and on the dmp branch of ROS Reality Bridge. Both of these repositories are

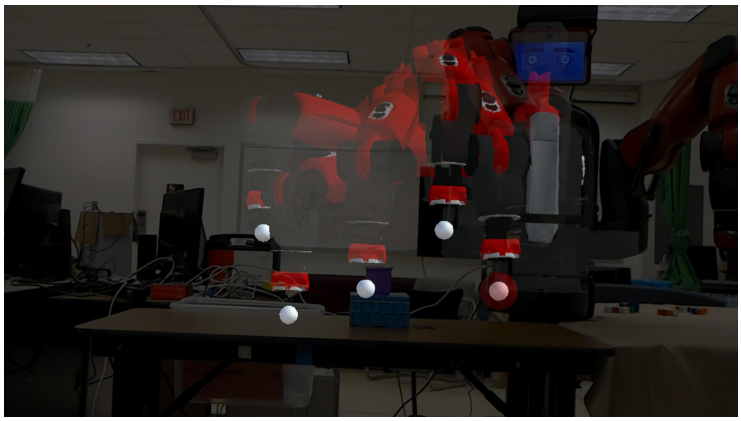


Figure 5: *The white spheres represent endpoints with separate DMPs in between them. This figure is the actualization of* ***Figure 3*** *after a similar training scenario. Shadows are shown over the spheres to give an idea of the orientation of the arm.*

located on the H2R Github [17, 18]. We use ROS Reality Bridge [19, 20], which allows our Unity app to Advertise, Publish, and Subscribe to ROS topics. We now give a description of the architecture, summarized in **Figure 6**.

4.1. HoloControl

The HoloControl Application has five main components: the shadow mesh, WebsocketClient, TFListener, SpeechManager, and GestureManager.

The Unity rendering of the shadow is created by parsing a Unified Robot Description Format (URDF) of the Baxter. The WebsocketClient connects to the ROS Reality Bridge Server, which is an interface to the Baxter. The WebsocketClient marshals data and sends it to the bridge. The bridge publishes this information to the desired ROS topic. The bridge also forwards all messages via the Unity Node. The TFListener uses the WebscocketClient to subscribe to a transforms topic, which the Baxter publishes to when its joints move. When there is an update, the TFListener updates the shadow.

The SpeechManager is used to control the flow of the application and is critical for starting up the processes of collecting data and moving autonomously. It responds to various voice commands detailed in sections 3.3–3.5. Using the WebsocketClient, the SpeechManager publishes commands interpreted by the DMP Node and the Inverse Kinematics (IK) Interface running on the bridge. The SpeechManager sends commands to the DMP node to record arm state, create a data log for a new skill, and execute motion plans. The SpeechManager sends communicates with the IK interface to make the Baxter arm gripper open and close.

The GestureManager also sends messages used in both the DMP Node and the IK Interface. During normal teleopperation, every time the sphere is moved, its coordinates in R are sent to the IK Interface to update the arm. When HoloControl is in recording mode to collect a demonstration, messages are sent to the DMP node to record a trajectory and to save CPs as the user clicks.

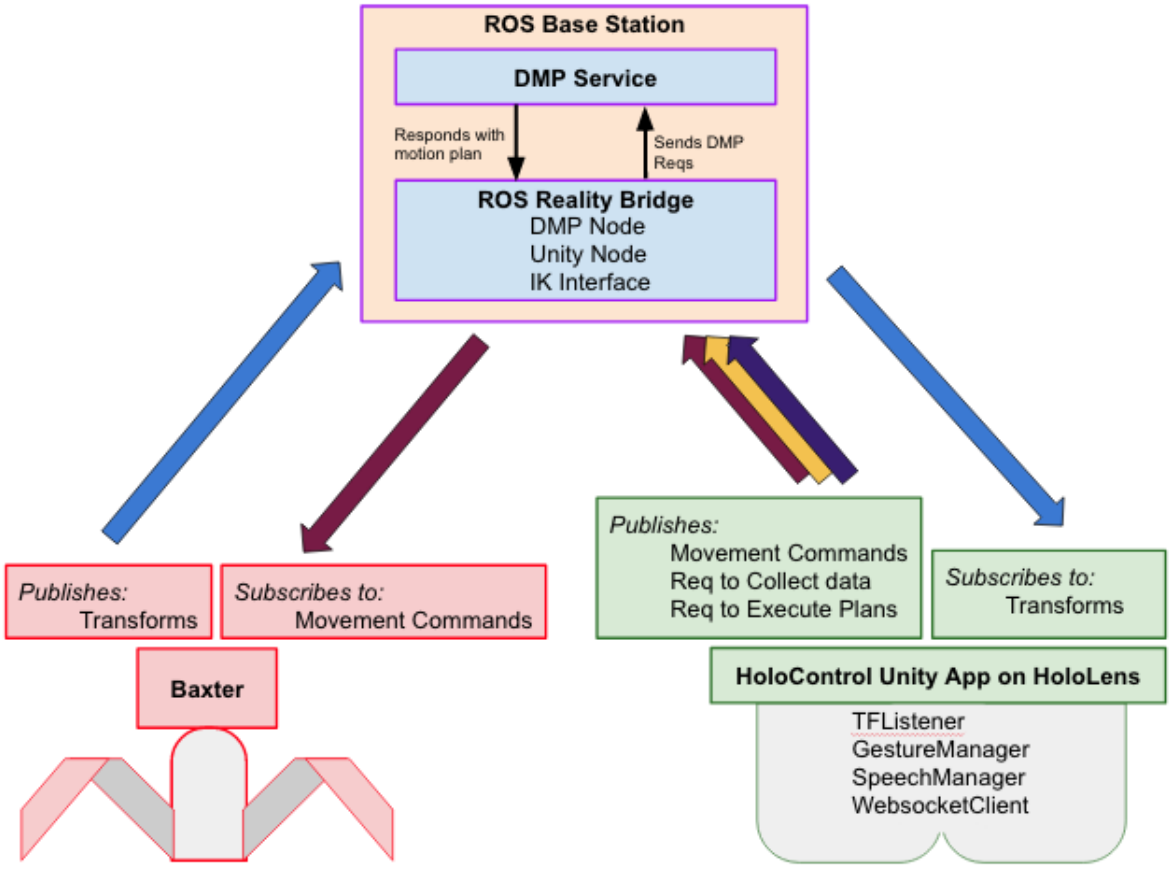


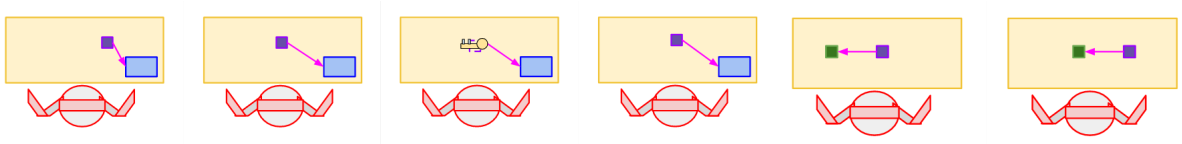
Figure 6: *During runtime, the system relies on communication between the Baxter robot and the HoloControl client via ROS Reality Bridge. All components are connected to the same network. ROS Reality Bridge is also responsible for making LfD request and handling data logging upon request from the client. It forwards all movement commands sent by the client to Baxter.*

4.2. DMP Node

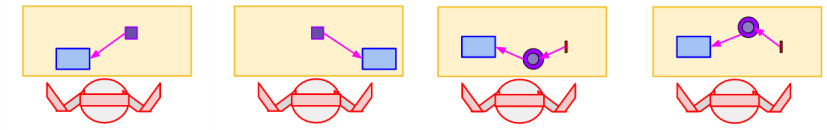
The DMP Node responds to requests related to collecting demonstration data: end-effector position, gripper state, and CPs. When a skill is completed, it uses a data filter to fit a biased least square model through the skill data, as supported by the CPs. It then makes a request to the ROS DMP service [16] to create a motion plan for the skill. After training and user parameter manipulation, the DMP Node executes each of the skill plans and publishes the resulting end-effect positions to move the arm autonomously.

5. Demos

To highlight features of our system, we follow a similar experimental model to that of Elliott et. al. [11]. We evaluated our MR LfD system on five pick and place tasks shown in **Figure 7**. Our tasks determine the extent to which the manipulator could behave autonomously after training. A summary of our results can be found in **Table 1**.



(a) *The pick item is displaced.* **(b)** *The pick item is swapped.* **(c)** *Both pick and place locations displaced for a fine-grain motion.*

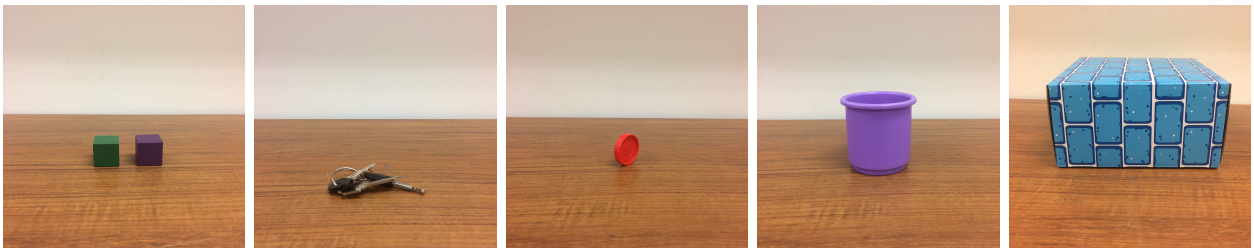


(d) *The place location is displaced.* **(e)** *Chained demonstrations are reconfigured.*

Figure 7: *Diagrams showing our various task. For each task,* ***(a)-(e)****, the right image represents the training objective and the left image shows the testing goal objective. An arrow starting at an object signifies that we must pick up this item. An arrow terminating on an object signifies that a picked item should be placed on it.*

5.1. Experimental Setup

For our pick and place tasks, we work with wooden cubes that have side length 2.54 cm, house keys that have maximum length 6 cm, a plastic coin with diameter 3.2 cm and thickness 0.3 cm, and a plastic cup with external diameter, height 7.62 cm and rim thickness 0.6 cm. We also use a 15.24 cm x 15.24 cm x 7.62 cm box as a raised square surface on which to place items. There is no stipulation about where on the box objects must be placed. **Figure 8** shows our objects.



(a) *Cubes* **(b)** *Keys* **(c)** *Coin* **(d)** *Cup* **(e)** *Box*

Figure 8: ***(a)-(e)*** *show the various objects used for our demo.*

For a given task, an expert demonstrator was asked to teleoperate the Baxter arm to achieve a training objective. After this initial demonstration, the user was asked to adapt the DMP segments to achieve a test objective. The train time (excluding calibration time), number of skills, adaptation time, and success rate of picks and places were recorded.

5.2. Task Descriptions

(a): *Training*: A cube is in front of the Baxter and must be placed on the box to the right of the cube. The cube must make contact with the box before it is released. Tape of thickness 2.54 cm was placed around the cube to make sure it could be returned to the same position. *Testing*: The cube is moved 10 cm to the right, and hence closer to the box. The goal is still to place the cube on the box.

(b): *Training*: In this case the training setup is the same as that of **(a)**. *Testing*: The cube is replaced with three house keys on a ring. The location of the keys is roughly the same as that of the cube from training time.

(c): *Training*: Two cubes are placed 7.6 cm apart and in front of the Baxter. The rightmost cube must be picked and placed on top of the second cube. *Testing*: Both cubes are moved left by 5 cm and the right most cube is again to be placed on the second cube. The placement was considered a failure if the gripper knocked off the top block even after placing it.

(d): *Training*: In this case the training setup is the same as that of **(a)**. *Testing*: The position of the cube does not change. However, the box is moved to the left of the cube and hence the place site changes.

(e): *Training*: A plastic coin and box are positioned in line; a cup is positioned in between them. The cup is at an angle with respect to the coin and the box such that it is farther away from the Baxter. The coin must be picked and dropped (not necessarily placed) in the cup. The cup must then be picked and placed on the box. *Testing*: The cup now forms an angle with the coin and the box such that it is closer to the Baxter. The sequence of sub-goals within the demonstration remains the same.

5.3. Results and Discussion

(a): The training task took an average of 55.3 s over three trials using our system. In all three trials, the demonstration used three skills, with two CPs in the first and second skill and none in the third. In the first skill the trainer centered the end-effector over the cube, lowered the arm, and then closed the gripper around the cube. In the second skill, the trainer moved the arm over the box and then more slowly positioned the cube on the box. In the third skill, the arm was moved away from the place site.

During the tests, the user took an average of 18.9 s to move the endpoint spheres. The trainer noted that it was useful to have the tape surrounding the cube as a landmark to deal with calibration issues. The arm successfully picked up and placed the cube all three times, following the general motion from the training.

(b): The training time averaged 55.3 s, which was slightly faster than that of **(a)**. This is perhaps because the trainer had already completed three trials of this training task to complete **(a)**. The motions in these training trials looked like those in **(a)** with the same distribution of skills and CPs.

The user opted not to move the endpoints, as the key was in roughly the same location as the cube. The keys were picked up successfully twice and placed on the box once. Due to the irregular nature of the keys, it was sometimes hard for the Baxter arms to pick up the keys. There were two different failure modes that were observed when placing the key on the box. First, when the gripper was not making a lot of contact with the keys, a slight jerk resulted in the keys slipping. Second, the keys were placed in such a way that they slipped off of the box. We believe that building a more intelligent system, which works based on visual feedback, and not just DMP endpoints will be more successful at this task.

(c): It took an average of 55.0 s to complete training. Because of the precise nature of this task, much of the time as spend placing the cube on top of the other cube. The user again opted to use 2 CPs for the pick up skill and 2 CPs for the place skill. No CPs were specified when moving the gripper away from the place site. During the first skill, the user consistently opted to reach a high position over the cube and then go downwards. After the pick, the end-effector was also move upwards to avoid hitting the adjacent place cube.

An average of 38.7 s were used to move the two spheres defining the pick site at the first cube and the place site at the second cube. The user again noted that the tape around the blocks was a critical reference point for dealing with calibration issues. The pick task was successful all three times. However, the place task was never successful. We noticed two failure modes. In the first the cube was placed at an offset and hence fell off immediately. In the second, the cube was placed correctly initially, however, the process of the gripper moving away knocked the cube over.

(d): The training time took an average of 39.5 s. The motions in these training trials looked like those in **(a)** with the same distribution of skills and CPs. We note the apparent time benefit associated with frequently using our system.

Adjustment time was also quick and took an average of 17.5 s. The user moved the

place point and the end point of the demonstration, but not the start point. The pick was successful all three times, however, the place failed once. This failure mode occurred because the arm arced downwards, instead of upwards as in the demonstration, and hence made contact with the box before reaching the place site. This result seems to support the fact that DMP plans can be more error prone if start and end points are shifted significantly.

(e): The training task took 72.5 s on average. The user specified 2 CPs en route to pick up the coin and 2 more in getting above the cup. After the drop no CPs were specified when motioning to grip the cup. After this pick, two more CPs were specified in a large arc that put the cup in position on the box. After the cup was released, no CPs were specified to move the end-effector away from the cup. This pattern was consistent on all three trials.

Adjustment time took 32.0 s on average, and was focused on the drop point for the coin and the pick point for the cup. The coin was picked up all three times, which was expected considering this skill was essentially a replay of the demonstration. The drop and grab were also successful all three times. The place of the cup was successful only two out of three times. The failure mode was similar to that of **(d)**, in that the arced path ended up approaching the cup place point from the bottom as opposed to the top.

Task Letter	Avg. Train Time [s]	Std. Train Time [s]	Num. Skills	Avg. Added Test Time [s]	Std. Added Test Time [s]	Frac. Picks	Frac. Place
a	55.3	9.0	3	18.9	6.8	3/3	3/3
b	50.1	4.8	3	0.0	0.0	2/3	1/3
c	55.0	4.6	3	38.7	7.1	3/3	0/3
d	39.5	2.3	3	17.5	2.9	2/3	3/3
e	72.5	1.7	5	32.0	3.6	6/6	5/6

Table 1: *Results across the five tasks, showing the relative strengths and weaknesses of the system.*

6. Future Work

While our system is a good prototype of an MR LfD system, there is rigorous work to be done to support this claim. In all scenarios, the robot was trained by an expert demonstrator, who was comfortable with the system and had used it over 100 times before any results were collected. Furthermore, there was a single teacher for all demos. We plan to do a user study to collect further statistics on the amount of time that a novice user must spend in order to complete a set of task. We also plan to address a slew of concerns that will make our system more usable and robust to user error.

The biggest pitfall of our system is calibration. Because of the lack of stability in holograms, there is often drift as the user moves around the robot or looks away. The holographic and real-world spaces do not always line up. It becomes hard to move the

DMP parameter spheres to precise locations. To mitigate against these sources of error, we propose using Vuforia AR tags and HoloLens spatial anchors to better fix the location of the shadow relative to the real robot.

We also plan to use the Baxter wrist camera to implement basic object recognition on items that are picked up. This way we can learn a mapping from objects to actions such that the actions center the object in our field of view. Even if the location a user specifies in MR is off by some factor, the manipulator should be able to recognize this case and course correct autonomously.

We also note that we are not giving the user control over all degrees of freedom of the Baxter arm. For example, the user cannot change the orientation of the gripper. While this simplification makes our interface less complicated, it is worth exploring if giving control of more degrees of freedom would improve the range of tasks or reduce the amount of time per task.

Our system allows a teacher to easily convey information to the robot. However, it does not allow for the robot to covey its knowledge to the user. For example, the IK solver on the Baxter works based on the current position of the arm. Hence the arm must actually move to use the solver. However, if our application used an IK solver in a Baxter Gazebo environment, we could update the shadow without moving the actual robot arm. Furthermore, we could play out motion plans for the user before they are executed. If a user were dissatisfied, then they should be able to tweak the model, perhaps by adding or changing CPs.

It is also worth comparing our approach to one that just makes a strong key-frame assumption: the path between CPs does not matter and can be linear. In this case there is no explicit need for learning. The user must only provide a space set positions, specified via MR, and the action to be taken at each point. Assuming the functionality of being able to play back motion plans in MR and specify more points an needed, this version of the system would help test the extent to which full demonstrations are necessary.

On an exciting note, our combination of MR DMP approach is not limited to the Baxter arm. We plan to adapt HoloControl as an interface to other ROS enabled devices to bring quadcopters and mobile manipulators into MR.

7. Conclusion

We propose a MR LfD system that uses holograms to control robot action. Our system serves as a simple interface to the robot and supports touch-free teleoperation of the robot arm. In our system, a user places a MR hologram of the robot on top of the real robot. The position of hologram is maintained by a HoloLens HMD. The user then moves a holographic sphere that appears on the robot end-effector and hence moves the real end-effectors. The user collects data for—possibly—many skills within a demonstration. A different motion plan is created for each skill. Our system also supports CPs, which are key positions that are integral to the success of the demonstration. Data is processed using a bias least fit method, in which CPs are weighted more heavily. Separate DMPs

are learned for each skill based on the filtered data. The user can then edit the endpoints of the skills and execute the motion plans.

The main advantages of our system are that it does not involve making contact with the robot, forces the user to pay attention to the robot, allows the user to quickly generalize a demonstration, and effectively chains many skills together.

Acknowledgements

Big thank you to George for taking me as an advisee last minute. I am grateful for this experience and hope to work with you in the future. Thank you Stefanie for being an incredible mentor. Your passion for robotics is infectious. Thank you David and Eric for sharing your gear and helping me when I was stuck. This thesis truly would not have been possible without you both.

I would also like to thank my family and friends who have been there for me throughout college. Thank you Fiona Stolorz for all the hours you kept me company in the lab. Your support has meant the world to me. Thank you Mickey Zaslavsky for keeping me in good spirits. Thank you Anthony Daoud for being my closest friend.

References

[1] Argall, B.; Chernova, S.; Veloso, M. and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* (Vol. 57, Issue 5) 469-483.

[2] Chernova, S. and Andrea, L.T. 2014. Robots Learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* (Vol. 8).

[3] Mülling, K.; Kober, J.; Kroemer, O. and Peters, J. 2013. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research* (Vol. 32, Issue 3) 263-279.

[4] Pilarski, P.M.; Dick, T.B. and Sutton, R.S. 2013. Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints. *IEEE International Conference on Rehabilitation Robotics* 1-8.

[5] Hung, P. N. and Yoshimi, Takashi. 2016. An approach to learn hand movements for robot actions from human demonstrations. *IEEE/SICE International Symposium on System Integration*.

[6] Vasan, G. and Pilarski, P.M. 2017. Learning from demonstration: Teaching a myoelectric prosthesis with an intact limb via reinforcement learning. *International Conference on Rehabilitation Robotics* 1457-1464.

[7] Ijspeert, A.J.; Nakanishi, J.; Hoffmann, H.; Pastor, P. and Schaal, S. 2013. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation* (Vol. 25, Issue 2) 328-373.

[8] Pastor, P.; Hoffmann, H.; Asfour, T. and Schaal, S. 2009 Learning and generalization of motor skills by learning from demonstration. *IEEE International Conference on Robotics and Automation* 763-768.

[9] Bishop, C. 2006. *Pattern recognition and machine learning*. Springer.

[10] Akgun, B.; Cakmak, M.; Jiang, K. and Thomaz, A. 2012. Keyframe-based learning from demonstration. *International Journal of Social Robotics*. (Vol. 4, Issue 4) 343-355.

[11] Elliott, S.; Toris, R. and Cakmak, M. 2017. Efficient Programming of Manipulation Tasks by Demonstration and Adaptation. *Robot and Human Interactive Communication* (26).

[12] Chong, J.; Ong, S.; Nee, A. and Youcef-Youmi, K. 2008. Robot programming using augmented reality: An interactive method for planning collision-free paths. *Robotics and Computer-Integrated Manufacturing* (25) 689-701.

[13] Ong, S.; Chong, J. and Nee, A. 2010. A novel ar-based robot programming and path planning methodology. *Robotics and Computer-Integrated Manufacturing* (26) 240-249.

[14] Fang, H.; Ong, S. and Nee, A. 2012. Interactive robot trajectory planning and simulation using augmented reality. *Robotics and Computer-Integrated Manufacturing* (28) 227-237.

[15] Rosen, E.; Whitney, D., Phillips, E.; Chien, G.; Tompkin, J.; Konidaris, G. and Tellex, S. 2017. Communicating robot arm motion intent through mixed reality head-mounted displays. *Computing Research Repository*.

[16] ROS DMP. `http://wiki.ros.org/dmp`.

[17] Holobot: Baxter DMP Branch. `https://github.com/h2r/Holobot/tree/holocontrol_baxter`.

[18] ROS Reality Bridge, DMP Branch. `https://github.com/h2r/ros_reality_bridge/tree/dmp`.

[19] Crick, C.; Jay, G.; Osentoski, S. and Jenkins, O. 2012. ROS and rosbridge *Human-Robot Interaction* 493-494.

[20] Whitney, D.; Rosen, E.; Phillips, E.; Konidaris, G. and Tellex, S. 2017. Comparing Robot Grasping Teleoperation across Desktop and Virtual Reality with ROS Reality. *Robotics Research: the 17th Annual Symposium*.