Generalizing Learned Manipulation Skills in Practice

Juan Wilches, Yongqiang Huang, and Yu Sun

Abstract— **Robots should be able to learn and perform a manipulation task across different settings. This paper presents an approach that learns an RNN-based manipulation skill model from demonstrations and then generalizes the learned skill in new settings. The manipulation skill model learned from demonstrations in an initial set of setting performs well in those settings and similar ones. However, the model may perform poorly in a novel setting that is significantly different from the learned settings. Therefore a novel approach called generalization in practice (GiP) is developed to tackle this critical problem. In this approach, the robot practices in the new setting to obtain new training data and refine the learned skill using the new data to gradually improve the learned skill model. The proposed approach has been implemented for one type of manipulation task – pouring that is the most performed manipulation in cooking applications. The presented approach enables a pouring robot to pour gracefully like a person in terms of speed and accuracy in learned setups and gradually improve the pouring performance in novel setups after several practices.**

I. Introduction

A. Background

The reason we can perform everyday tasks in various settings is that we can generalize learned skills. Robots should have the ability to learn and perform a manipulation task across different settings as well. A primary goal in designing a learning approach for manipulation skills is to guarantee that the learned skill model will generalize - perform accurately in new settings after being trained in a finite number of settings.

Analytical models can generalize well if they are based on physics laws and have all necessary variables and coefficients. However, these models are hard to define for manipulation tasks since it is unrealistic to obtain all the variables and their accurate coefficients, especially when dealing with fluid and soft objects. Neural network models such as a recurrent neural network (RNN) [1], [2] can learn complex dynamics, but may not generalize well to data outside of the training distribution. Regularization techniques such as dropout can reduce the model complexity and thereby achieve a better generalization. In [3], the authors provide an insightful discussion on explicit and implicit regularization techniques and their effects on improving the generalization of neural networks.

Generalization of a neural network does not depend on only its architecture and training techniques, but also the training dataset. For manipulation tasks, the training samples must be sufficiently large and come from a vast of different settings. However, before we have a physics simulator that can accurately simulate dynamics for fluid and soft objects, manipulation training samples will remain costly to obtain. So, it is unrealistic to obtain large training samples in a vast of settings.

Inspired by the human ability to transfer the manipulation skills learned in a small number of settings to a new setting after several practices, we have developed an approach called Generalization in Practice (GiP) for a robot to generalize the learned manipulation skills. Our approach first learns a skill model in several settings. Then a robot practices the learned skill in a new setting that is very different from the training settings. The model is refined using the data collected during the practices in the new setting with one trick that is when training the model, the desired outcomes in the data are replaced by the real outcomes of the manipulation during the practices. Switching out the desired outcome with the real outcomes in training makes the neural network re-calibrate to the new setting.

To demonstrate the approach, we choose pouring water as the manipulation task to evaluate the proposed approach because not only pouring is the most performed manipulation in cooking applications [4], but also water is difficult to model and control [5]. It is especially challenging for a robot to achieve accurate pouring, a problem that cannot be solved using traditional control policies for two reasons:

1) Lack of precise dynamics models: modeling fluid or granular motion precisely is either impossible or unfeasible because there are many unobservable parameters, and those parameters vary with many factors such as the material and the shape of the pouring device.
2) Un-reversible feature of the task: poured material cannot come back to the pouring device once it is poured out. Therefore there cannot be overshoot in the system's response.

The two difficulties go hand-in-hand. The irreversibility of pouring calls for a predictive approach that can predict when to stop and rotate back the pouring container to avoid over-pouring.

B. Related Works

In order to teach a robot manipulation skills, researchers have developed many learning-from-demonstration (LfD) approaches [6], in which a teacher demonstrates how to perform a task, a policy is derived from the demonstration and then transferred to a robot [7]. Several works have been published related to robot learning manipulation skills by using LfD. In [8] the authors applied reinforcement learning to learn how to turn a valve and grasp a bottle from a table.

The authors are members of the Robot Perception and Action Lab (RPAL) in the Department of Computer Science & Engineering at the University of South Florida, Tampa, FL, USA. {jwilches,yongqiang,yusun}@mail.usf.edu

In [9], a robot learned to place, push, and pick-and-place a new object from a demonstration of a single video. Other successful LfD applications include playing table tennis [10], grasping and using tools [11]–[13], and many more [14]–[16].

Recent works have proposed approaches for the accurate pouring of liquids. The approaches in [17] and [18] used RGB-D cameras to measure the water height in the receiving container and control the rotation of the pouring container with a simple PID controller. Their smallest mean errors were 38 milliliters (ml) and 13.2 ml, respectively. However, those methodologies start the container's backward rotation when the target height is reached, a technique that might lead to over-pouring since liquid still comes out when the backward rotation starts. The approach in [19] used reinforcement learning with a water simulator to learn a policy for accurately pouring water and transfer the policy to the robot, which had a mean error of 19.96 ml. The authors also used RGB-D cameras to detect liquid height. In [20] the authors rely on an audio spectrogram to determine the volume poured by the robot. The mean volume errors reported for different receiving containers ranged from 6.42 ml to 13.79 ml, such small errors are achieved by the usage of a spout at the opening of the pouring containers, which reduces the speed of pouring.

In [2], the authors propose predictive model control (MPC) for accurate robotic pouring in which they achieved mean volume errors ranging from 7.25 ml to 26.13 ml for different pouring containers. Another work [1] presented a Long-Short-Term Memory (LSTM) model that was trained using demonstration data. However, the learned model was only evaluated in simulation. In [21] the authors use both vision and weight, achieving a pouring error of less than 5 ml with a time from 20 to 45 seconds per pour. In our dataset, collected from human demonstrations, the pouring time ranged from 3.2 to 8.7 seconds to pour water. If requested, humans can also achieve a small pouring error by pouring slowly.

II. Learning Manipulation Skill

To learn a manipulation skill, we first define a general time-sequence model as

$$\mathbf{y}(t) = \mathbf{F}(\mathbf{y}(1, ..., t-1), \mathbf{o}(1, ..., t-1), \mathbf{u}) \tag{1}$$

where $\mathbf{y}(t)$ represents the manipulation motion signal such as velocity at time t; $\mathbf{y}(1, ..., t-1)$ and $\mathbf{o}(1, ..., t-1)$ are the motion signals and the outcomes respectively of all previous times, while $\mathbf{u}$ denotes the setting and the desired outcome. $\mathbf{F}(\cdot)$ is a non-linear function. The model gives a general representation of a manipulation skill where the motion signal at the current time is a function of the setting, the desired outcome, the outcomes of all previous time steps, and the motion signals of all previous signal steps.

Using pouring as an example, the current pouring rotation velocity is a function of the setting – the pouring cup size, the condition – initial water volume in the pouring cup, and the goal – the desired volume in the receiving cup, and states including the water volumes in the receiving cup of all previous steps (the outcomes of all previous time steps) and all previous pouring rotation velocities (the motion signals of all previous signal steps). Therefore, we can represent the general time-sequence model from Eq. (1) with:

$$\omega(t) = \mathbf{F}\left(\omega(\tau)_{\tau=1}^{t-1}, v(\tau)_{\tau=1}^{t-1}, h, \kappa, v_{initial}, v_{desired}\right) \tag{2}$$

where $\omega(t)$ is the angular velocity of the pouring container at time step t, $\omega(\tau)$ are the pouring container's velocities at all previous time steps, $v(\tau)$ are the volumes of water in the receiving container at at all previous time steps, h is the height of the pouring container, $\kappa = 2/d$ is the body curvature of the pouring container with d being the diameter of the container, $v_{initial}$ is the initial amount of water present in the pouring container before pouring, and $v_{desired}$ is the desired water amount should be poured into the receiving container. $\mathbf{F}(\cdot)$ represents a dynamical system that establishes relationship among the motion signals, the outcomes and the settings. For pouring, the angular velocity $\omega(t)$ is the action that pushes the motion forward.

Based on the time-sequence features of the manipulation skill model, we selected RNN as the skill model's structure since it is capable of processing time series inputs and has been shown to be an universal approximator of dynamic systems [22]. At each time, an RNN model takes in the current input and the hidden state of the RNN model at the previous time. The current outcome of the manipulation motion can be used as the input of the RNN model, while the past outcomes and motion signals can be represented in the hidden state of the RNN model at the previous time. The setting and the desired outcome can be provided to the RNN model either as the initial hidden state or as a part of the input. Based on our previous work [1], we designed the learning model using peephole long short-term memory (LSTM) units [23]. The inputs and outputs are processed inside the peephole LSTM cell for each time step t as follows:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{y}_{t-1} + \mathbf{b}_i + \mathbf{p}_i \odot \mathbf{c}_{t-1}\right) \tag{3}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{y}_{t-1} + \mathbf{b}_f + \mathbf{p}_f \odot \mathbf{c}_{t-1}\right) \tag{4}$$

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_g\mathbf{x}_t + \mathbf{U}_g\mathbf{y}_{t-1} + \mathbf{b}_g\right) \tag{5}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{6}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{y}_{t-1} + \mathbf{b}_o + \mathbf{p}_o \odot \mathbf{c}_t\right) \tag{7}$$

$$\mathbf{y}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{8}$$

where $\mathbf{i}_t$, $\mathbf{o}_t$, and $\mathbf{f}_t$ are the input, output, and forget gates respectively. The matrices ($\mathbf{W}_*$,$\mathbf{U}_*$,$\mathbf{b}_*$,$\mathbf{p}_*$) represent the weights, biases and peephole connections weights to be learned respectively. The sequences $\mathbf{x}_t$, $\mathbf{c}_t$, and $\mathbf{y}_t$ are the input, cell state and hidden state of the network at time step t respectively. σ represents the sigmoid function that controls the information flow through the gates. "tanh" represents the tanh function that is used to alleviate the vanishing or exploding gradients. $\odot$ represent element-wise multiplication. The illustration of the peephole LSTM cell is shown in Fig. 1.

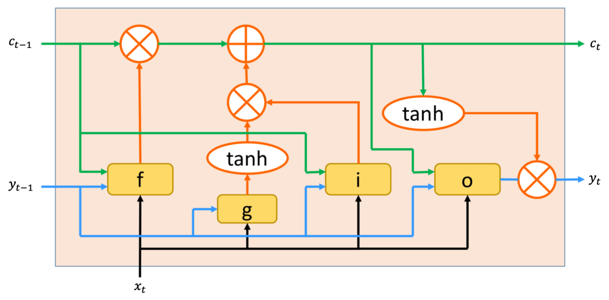


Fig. 1: Mechanism of a peephole LSTM cell

From Eq. (2) we established the inputs to the network at each time step as:

$$\mathbf{x}_t = [\theta_t, v_t, h, \kappa, v_{initial}, v_{desired}] \tag{9}$$

where θ_t is the pouring container's angle at time step t that represents the numerical integration of the velocities of all previous time steps, v_t is the volume of water in the receiving container at time step t that represents the outcome volume of all previous time steps. The rest of the features are the same as in Eq. (2). The inputs can be represented as:

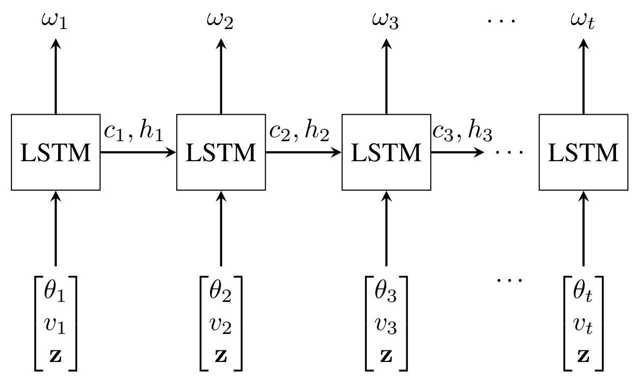
$$\mathbf{x}_t = [\theta_t, v_t, \mathbf{z}]^\top \tag{10}$$

where $\mathbf{z} = [h, \kappa, v_{initial}, v_{desired}]$ represent the setting of the pouring task that distinguishes one task from another. The variables θ_t and v_t represent the change of the dynamics of the system.

The output of the network is:

$$\omega_t = \mathbf{W}_\omega \mathbf{y}_t + \mathbf{b}_\omega \tag{11}$$

where ω_t is the angular velocity of the pouring container at time step t, $\mathbf{W}_\omega$ and $\mathbf{b}_\omega$ represent the weights of a fully connected layer that reduces the vector $\mathbf{y}_t$ to a scalar. The processing of the recurrence of inputs and outputs is shown in Fig. 2.



ω_1
ω_2
ω_3
$\cdots$
ω_t
LSTM
c_1, h_1
LSTM
c_2, h_2
LSTM
c_3, h_3
$\cdots$
LSTM
$\begin{bmatrix}\theta_1\\ v_1\\ \mathbf{z}\end{bmatrix}$
$\begin{bmatrix}\theta_2\\ v_2\\ \mathbf{z}\end{bmatrix}$
$\begin{bmatrix}\theta_3\\ v_3\\ \mathbf{z}\end{bmatrix}$
$\cdots$
$\begin{bmatrix}\theta_t\\ v_t\\ \mathbf{z}\end{bmatrix}$

Fig. 2: High level view of the LSTM model inputs and outputs with $\mathbf{z} = [h, \kappa, v_{initial}, v_{desired}]^\top$

Training. The LSTM RNN skill model was trained with pouring data in the open Daily Interactive Manipulation (DIM) dataset [24], [25]. The DIM dataset does not have a direct volume reading of the receiving container. Instead, it has the recording of a force sensor under the receiving container. The volume is proportional to the weight reading. The dataset has 284 pouring trials of 9 containers, from which 221 were used for training and 63 for validation. The loss function in the training is defined as the mean squared error between the recorded and predicted angular velocity ω_t:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i - 1} \sum_{t=1}^{T_i - 1} (\hat{\omega}_{it} - \omega_{it})^2. \tag{12}$$

where $\hat{\omega}_{it}$ and ω_{it} are the predicted angular velocity and actual angular velocity for trial i and time step t respectively, N is the number of trials, and T_i is the number of time steps of the trial. After trying and training different LSTMs with varying number of layers and units, the final skill model architecture is composed of 1 layer and 16 LSTM units. The number of units establish the vector size of Eqs. (3) to (8). The model was trained for 2000 epochs with 50% dropout and learning rate of 1×10^{-3}. The model with the lowest validation loss was selected as the pouring skill model. We refer to this network as model M_0 in the paper.

III. Generalization in Practice

As expected the trained manipulation skill model (model M_0) pours accurately with the same or similar pouring containers used in training as shown in the evaluation section IV-B. However, the model performs poorly when the setting is changed – using dramatically different containers as shown in Section IV-C. This requests the novel generalization in practice (GiP) approach.

Practice. When the robot is facing a new set – an unseen container that is significantly different from the ones in the learning set, the robot can performs pouring practices using the learned skill model M_0. The outcomes are observed and recorded. If the outcome is not satisfactory, such as the error of the poured amount is much higher than an averaged human pouring error, the robot knows the learned skill model M_0 does not work well in the new setting. However the data collected during practices are useful to fine-tune the skill model. If we replace the desired pouring goals with the actual pouring outcomes in the practice data, we can fine-tune the model using the practice data for the new setting.

Fine-tuning. The practice data for the fine-tuning have inputs including the new setting, **the goal – the actual outcome in practices**, initial conditions, and states. The output in the practice data would be the robot's current manipulation motion at each time step. For pouring manipulation and the LSTM model, he inputs of the data for fine-tuning are

- h: height of the pouring container.
- $\kappa = 2/d$: body curvature of the pouring container with d being the diameter of the container.
- $v_{initial}$: the initial amount of water present in the pouring container before pouring.
- $v_{desired}$: **the final water volume in the receiving container in the practice**, not the desired value provided to the robot for practice.

- v_t: the volume of water received in the receiving container **in the practice** at time t.
- θ_t: pouring container's current angle **in the practice** at time t.

For fine-tuning, the desired outcome is replaced by the actual outcome in the practice as indicated above in bold. The output of the fine-tuning is

- ω_t: the angular velocity of the pouring container **in the practice**.

Using the altered inputs and output, the data generated in practice can be used to fine-tune the skill model.

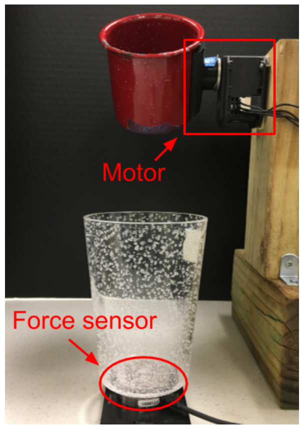
Gradual Fine-tuning. The practicing and fine-tuning (or transfer learning) can be carried out iteratively. However, in practice, one sample does not provide enough data for a stable fine-tuning. Instead, we fine-tune the skill model using a small number of practices in each iteration. The resulting new skill model is tested to verify whether its performance is better than the previous one with a new set of practices. If it is not, the new practices are added into the fine-tuning dataset. Then a new model is fine-tuned, and its performance is verified again. This iteration is done until a good enough performance is reached. Algorithm 1 describes this working mechanism with $n < 15$. We represent the performance of the skill model using an error whose lower value indicates better performance. The skill model is fine-tuned for several epochs equivalent to 10 times the number of available samples with 50% dropout and learning rate of 1×10^{-4}.

Batch Fine-tuning GiP can also be carried out in batch mode. The skill model is fine-tuned once for all after a large number of practising data has been accumulated. In this case only one fine-tuning is performed, instead of several fine-tunings in gradual fine-tuning mode. It is a result of running only one loop of Algorithm 1 with $n > 30$. However, this model could be risky and costly as the robot needs to practise the initial skill model for a large number of times and the performance in all the practices is poor. The skill model is fine-tuned with the same hyperparameters used for gradual fine-tuning.

The generalization in practice (GiP) with gradual fine-tuning is described in Algorithm 1.

Algorithm 1 Generalization in Practice (Gradual fine-tuning)

```
1: $M_{init} \leftarrow$ Initial model
2: $n \leftarrow$ Number of practices
3: $\mathcal{P} \leftarrow \{(p_s^1, p_d^1), ..., (p_s^n, p_d^n)\}$ A set of $n$ manipulation practices
4:                 ▷ $p_s^i \leftarrow$ start status; $p_d^i \leftarrow$ desired outcome
5: $err\_sa \leftarrow$ Satisfactory precision
6: $\mathcal{D} \leftarrow \{\}$
7: procedure PRACTISE($M$, $n$, $\mathcal{P}$)
8:     repeat
9:         Robot executes one practice in set $\mathcal{P}$, using model $M$
10:        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y)\}$   ▷ $(x, y)$ : input/output and outcomes of the practice are added into $\mathcal{D}$
11:    until $n$ practices are executed
12:    $error \leftarrow$ averaged error between actual outcomes and desired outcomes for all practices
13:    return $error$
14: end procedure
15: procedure GIP
16:    $M_{new} \leftarrow M_{init}$
17:    while True do
18:        $err \leftarrow$ PRACTISE($M_{new}$, $n$, $\mathcal{P}$)
19:        if $err < err\_sa$ then
20:            break
21:        else
22:            $M_{new} \leftarrow$ Fine-Tune($M_{new}, \mathcal{D}$)   ▷ Fine-tune the model using all practice data in $\mathcal{D}$
23:            $\mathcal{P} \leftarrow$ Generate-Random-Practices ($n$)   ▷ Randomly generate another set of practices
24:        end if
25:    end while
26:    return $M_{new}$
27: end procedure
```



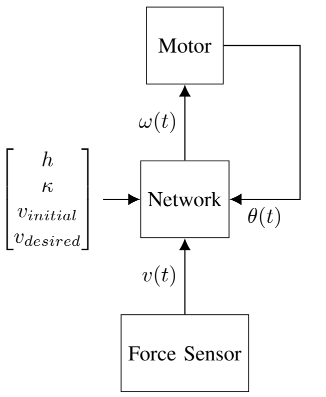


Fig. 3: Pouring robot architecture. The constant features $[h, \kappa, v_{initial}, v_{desired}]$ are sent to the network, the force sensor measures $v(t)$, the motor measures $\theta(t)$ and the network outputs $\omega(t)$. All features are handled by the network at each time step.

IV. Experiments and Evaluations

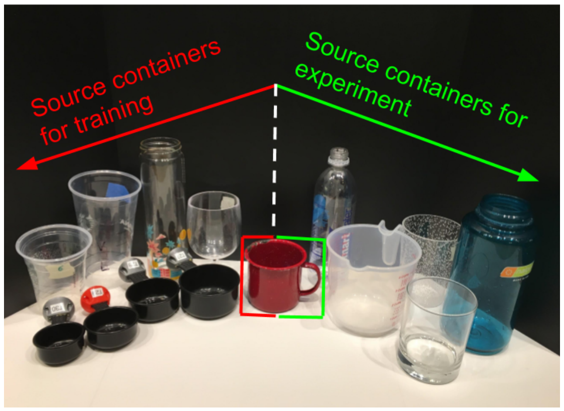
A. Pouring System

To evaluate the proposed approach, we have designed and developed a pouring robot. It consists of a Dynamixel MX-64 motor and ATI mini40 force-torque sensor. The motor was placed at a certain height above the surface with the pouring container attached to it and the force sensor was placed below the receiving container. Fig. 3 shows the configuration of the pouring system. This system works at 60 Hz which allows it to pour water at a pace similar to human pouring. We also tested our proposed approach in a UR5e collaborative robot arm.

B. Pouring Evaluations in Similar settings

We tested the accuracy of the pouring system for five different pouring containers not present in the training dataset.

The model used for these experiments is model M_0 that results from training using LfD. Fig. 4 shows the training and testing pouring containers where we can see that the Red Cup was used for training and testing. Table I summarizes the mean and standard deviation errors over 15 water pours performed by different robotic systems for each pouring container. The time taken for the system to perform the pouring motion ranged from 2.8 to 7.6 seconds. We can see that our system pours accurately to the testing cups and our mean errors are comparable to those encountered in the related works of accurate robotic pouring already discussed in section I-B. We can also see that the model worked for a UR5e collaborative robotic arm shown in Fig. 5 with a slight increase in pouring error [1].


Source containers for training
Source containers for experiment
Fig. 4: Cups used in the initial experiments.

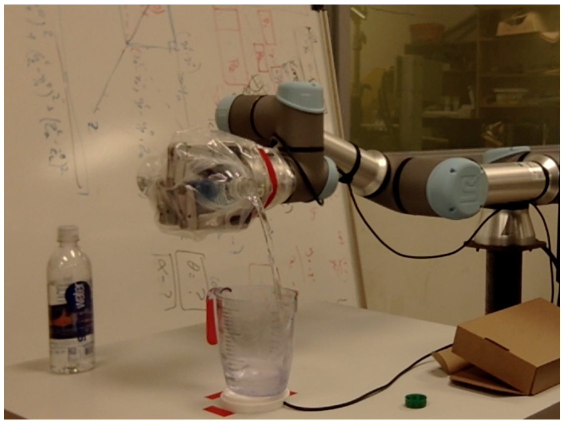


Fig. 5: Pouring example with UR5e collaborative robotic arm.

We compared our results with a baseline controller that applies a constant angular velocity for the forward and backward rotation. The goal of this experiment was to explore the difficulty of achieving accurate pouring using a simple controller. The forward constant velocity is applied to the pouring container until the target volume is reached, a similar approach used in [17], [18] but in our experiment we used a force sensor instead of vision to measure the volume of

[1]A video of pouring with the UR5e arm can be found at:
`https://youtu.be/xp9nEDTntU4`

TABLE I: Accuracy for pouring water from different pouring containers.

Robotic System	Pouring Container	μ_e (ml)	σ_e (ml)
Pouring Robot	Red Cup	4.78	3.56
	Water Bottle	4.12	4.29
	Bubble Cup	6.77	5.76
	Glass	7.32	8.24
	Tall Bottle	12.35	8.88
	Measuring Cup	13.13	8.04
UR5e robotic arm	Water Bottle	7.83	6.62

water in the receiving container. Then, the backward constant velocity is applied until the pouring container is in vertical position. The results can be seen in Table II for two pouring containers and baselines. The baseline ω_1 controller used 20 deg/sec as forward velocity and -30 deg/sec as backward velocity. The baseline ω_2 controller used 5 deg/sec as forward velocity and -7.5 deg/sec as backward velocity. We can see that when the forward angular velocity becomes smaller, the mean volume error decreases. This is expected since the pouring speed is decreased. However, we can also see that when the forward velocity is higher (similar to the human pace) the results of our model shown in Table I outperform the baselines' shown in Table II.

TABLE II: Results of Pouring with a Baseline Controller

Pouring Container	Baseline	μ_e (ml)	σ_e (ml)
Red Cup	ω_1	33.50	7.76
	ω_2	4.50	1.87
Bubble Cup	ω_1	56.25	5.85
	ω_2	22.25	4.29

C. Evaluations in Different settings

We aimed to evaluate model M_0 using new pouring containers, specifically a wine bottle and a blue bottle which have different geometry than the already trained and tested containers. Fig. 6 shows the scatter plot of height and diameter for the original training and testing containers and also for the wine bottle and blue bottle. We can see that these new containers are far in terms of height and diameter from the ones already used by the pouring system. In the evaluation, we kept all the other factors (initial volume, desired volume for each trial) the same to give a fair comparison.

Table III shows the mean and standard deviation error of the desired volume versus the actual volume for 15 pouring motions executed by the system. The system is inaccurate for pouring from the wine bottle and blue bottle more than ten times in average than pouring from the red cup. It is also inaccurate for pouring from the measuring cup around three times in average than pouring from the red cup.

D. Evaluating GiP

We evaluated our approach of GiP using batch fine-tuning for the wine bottle, blue bottle and measuring cup. The gradual fine-tuning approach was tested for the wine bottle as well. We tested the accuracy of the system by pouring 15

TABLE III: Comparison of accuracy for Red Cup, Wine Bottle, Blue Bottle and Measuring Cup of M_0.

Pouring Container	LSTM Model	μ_e (ml)	σ_e (ml)
Red Cup	M_0	4.78	3.56
Wine Bottle	M_0	51.22	39.61
Blue Bottle	M_0	55.85	47.32
Measuring Cup	M_0	13.13	8.04

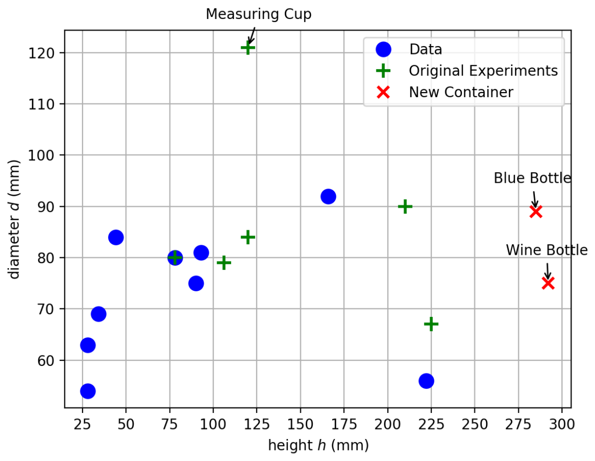


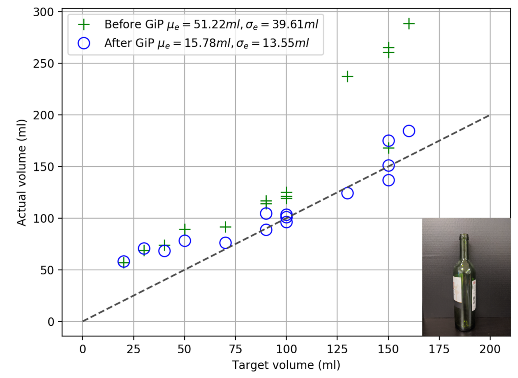
Fig. 6: Scatter plot of height vs diameter for pouring containers.

times per experiment for batch fine-tuning and carried out the experiments maintaining the same sets of volumes for fair comparison.

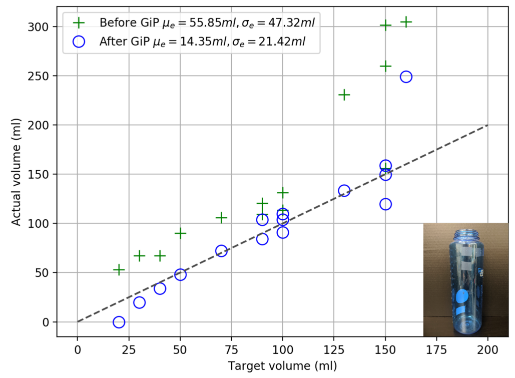
1) Wine Bottle: We collected a total of 36 pouring motions using the pouring system with the wine bottle as pouring container for different $v_{initial}$ and $v_{desired}$. Then, we fine-tuned the original LSTM RNN skill model using such dataset. Fig. 7(a) shows the mean and standard deviation errors for 15 pours using before and after fine-tuning M_0. Model M_1 was the result of applying GiP to M_0 using the wine bottle dataset. We can see that the wine bottle's mean error had a reduction of around 3 times compared with the initial 51.22 ml mean error. We can see that for model M_0 most of the trials resulted in over pouring. However, for model M_1 there are some trials that over pour but others under pour.

We also tested the gradual fine-tuning approach of GiP using the wine bottle. We decided to use it taking into account the high mean error it presents for pouring using the original LSTM model. We collected 10 pouring motions using the robot for each fine-tuning also for different $v_{initial}$ and $v_{desired}$. Table IV shows the evolution of the accuracy for the fine-tuning algorithms we carried out. The mean and standard deviation errors for this case are different from the ones shown in Table III as the $v_{initial}$, $v_{desired}$ and number of trials were different. We can see that after three fine-tuning runs the mean volume error is similar to that achieved by using batch fine-tuning.

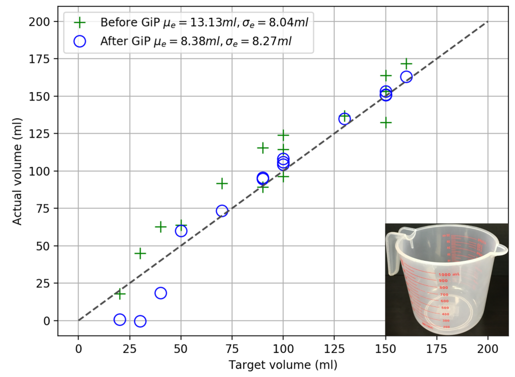
2) Blue Bottle: We decided to apply only batch fine-tuning for the blue bottle. We collected 54 pouring motions



(a) Wine Bottle M_0 *vs* M_1



(b) Blue Bottle M_0 *vs* M_5



(c) Measuring Cup M_0 *vs* M_6

Fig. 7: Accuracy comparison for applying GiP to different cups for the same $v_{initial}$ and $v_{desired}$.

TABLE IV: Accuracy for Wine Bottle after gradual fine-tuning.

Base Model	Fine-tuned Model	μ_e (ml)	σ_e (ml)
M_0		80.23	49.13
M_0	M_2	38.67	11.98
M_2	M_3	30.04	17.26
M_3	M_4	18.21	8.76

using the blue bottle as pouring container. Fig. 7(b) shows the mean and standard deviation errors for 15 pours using before and after fine-tuning M_0. Model Model M_5 was the result of applying GiP to M_0 using the blue bottle dataset. We can also see for this bottle the reduction in mean error.

3) Measuring Cup: We collected 36 pouring motions using the measuring cup. We also decided to use batch fine-tuning for this pouring container and applied GiP. M_6 results from applying GiP to M_0 using the measuring cup dataset. We can see a reduction again in mean error when comparing the desired and actual volume poured by the system.

V. Conclusion

In this paper, we presented a novel approach called generalization in practice (GiP) and demonstrated how it could be applied to accurate robotic pouring. The approach expands the generalization ability of a trained manipulation skill model to new settings. GiP considers the actual practice outcomes as the desired results and then uses them for training. We evaluated our approach with three pouring containers that are significantly different in geometry from the training containers. They were a wine bottle, a blue bottle, and a measuring cup that presented volume mean errors of 51.22 ml, 55.85 ml and 13.13 ml before applying GiP. Then, we applied GiP with batch fine-tuning, and the model achieved mean volume errors of 15.78 ml, 14.35 ml, and 8.38 ml for the same pouring containers, respectively. These results demonstrate that the proposed GiP approach can generalize learned manipulation skills to new settings. The GiP approach works for any applications, in which actual outcomes in practices can be used as the desired outcomes so that the actual outcome can substitute the desired outcome for training. In different manipulation tasks, outcomes could be a volume, a state [26], or a state change [27].

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. 1421418, No. 1560761 and No. 191004.

References

[1] Y. Huang, Y. Sun, Learning to pour, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 7005–7010. doi:10.1109/IROS.2017.8206626.

[2] T. Chen, Y. Huang, Y. Sun, Accurate pouring using model predictive control enabled by recurrent neural network, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 7688–7694. doi:10.1109/IROS40897.2019.8967802.

[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning requires rethinking generalization, arXiv preprint arXiv:1611.03530 (2016).

[4] D. Paulius, Y. Huang, J. Meloncon, Y. Sun, Manipulation motion taxonomy and coding for robots, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 5596–5601. doi:10.1109/IROS40897.2019.8967754.

[5] T. López Guevara, N. Taylor, M. Gutmann, S. Ramamoorthy, K. Subr, Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation, 2017, 1st Conference on Robot Learning 2017, CoRL 2017 ; Conference date: 13-11-2017 Through 15-11-2017. URL `http://www.robot-learning.org/`

[6] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Robot programming by demonstration, in: B. Siciliano, O. Khatib (Eds.), Handbook of Robotics, Springer, Secaucus, NJ, USA, 2008, pp. 1371–1394.

[7] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems 57 (5) (2009) 469 – 483. doi:https://doi.org/10.1016/j.robot.2008.10.024.

[8] A. Gupta, C. Eppner, S. Levine, P. Abbeel, Learning dexterous manipulation for a soft robotic hand from human demonstrations, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2016) 3786–3793.

[9] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, S. Levine, One-shot imitation from observing humans via domain-adaptive meta-learning, CoRR abs/1802.01557 (2018). arXiv:1802.01557. URL `http://arxiv.org/abs/1802.01557`

[10] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, A. G. Billard, Learning and reproduction of gestures by imitation, IEEE Robotics Automation Magazine 17 (2) (2010) 44–54. doi:10.1109/MRA.2010.936947.

[11] Y. Lin, Y. Sun, Grasp planning based on strategy extracted from demonstration, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 4458–4463.

[12] Y. Lin, Y. Sun, Robot grasp planning based on demonstrated grasp strategies, The International Journal of Robotics Research 34 (1) (2015) 26–42.

[13] Y. Lin, Y. Sun, Task-oriented grasp planning based on disturbance distribution, in: Robotics Research, Springer, 2016, pp. 577–592.

[14] Y. Huang, Y. Sun, Generating manipulation trajectory using motion harmonics, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 4949–4954. doi:10.1109/IROS.2015.7354073.

[15] D. Paulius, Y. Sun, A survey of knowledge representation in service robotics, Robotics and Autonomous Systems 118 (2019) 13–30.

[16] D. Paulius, N. Eales, Y. Sun, A motion taxonomy for manipulation embedding, in: Robotics: Science and Systems (RSS), 2020, pp. 1–9.

[17] C. Schenck, D. Fox, Visual closed-loop control for pouring liquids, 2017 IEEE International Conference on Robotics and Automation (ICRA) (2017) 2629–2636.

[18] C. Do, W. Burgard, Accurate Pouring with an Autonomous Robot Using an RGB-D Camera: Proceedings of the 15th International Conference IAS-15, 2019, pp. 210–221. doi:10.1007/978-3-030-01370-7_17.

[19] C. Do, C. Gordillo, W. Burgard, Learning to pour using deep deterministic policy gradients, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018) 3074–3079.

[20] H. Liang, S. Li, X. Ma, N. Hendrich, T. Gerkmann, F. Sun, J. Zhang, Making sense of audio vibration for liquid height estimation in robotic pouring, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 5333–5339. doi:10.1109/IROS40897.2019.8968303.

[21] M. Kennedy, K. Schmeckpeper, D. Thakur, C. Jiang, V. Kumar, K. Daniilidis, Autonomous precision pouring from unknown containers, IEEE Robotics and Automation Letters 4 (3) (2019) 2317–2324. doi:10.1109/LRA.2019.2902075.

[22] N. Mohajerin, S. L. Waslander, Multistep prediction of dynamic systems with recurrent neural networks, IEEE Transactions on Neural Networks and Learning Systems 30 (11) (2019) 3370–3383. doi:10.1109/TNNLS.2019.2891257.

[23] F. A. Gers, N. N. Schraudolph, J. Schmidhuber, Learning precise timing with lstm recurrent networks, J. Mach. Learn. Res. 3 (2003) 115–143. doi:10.1162/153244303768966139. URL `https://doi.org/10.1162/153244303768966139`

[24] Y. Huang, Y. Sun, A dataset of daily interactive manipulation, The International Journal of Robotics Research 38 (8) (2019) 879–886. doi:10.1177/0278364919849091.

[25] Y. Huang, M. Bianchi, M. Liarokapis, Y. Sun, Recent data sets on object manipulation: A survey, Big data 4 (4) (2016) 197–216.

[26] A. B. Jelodar, M. S. Salekin, Y. Sun, Identifying object states in cooking-related images, arXiv preprint arXiv:1805.06956 (2018).

[27] A. B. Jelodar, Y. Sun, Joint object and state recognition using language knowledge, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 3352–3356.