Human-guided Trajectory Adaptation for Tool Transfer

Robotics Track

Tesca Fitzgerald
Georgia Institute of Technology
Atlanta, GA
tesca.fitzgerald@gatech.edu

Elaine Short
University of Texas at Austin
Austin, TX
elaine.short@utexas.edu

Ashok Goel
Georgia Institute of Technology
Atlanta, GA
goel@cc.gatech.edu

Andrea Thomaz
University of Texas at Austin
Austin, TX
athomaz@utexas.edu

ABSTRACT

We introduce "transfer by correction": a method for transferring a robot's tool-based task models to use unfamiliar tools. By having the robot receive corrections from a human teacher when repeating a known task with a new tool, it can learn the relationship between the two tools, allowing it to transfer additional tasks learned with the original tool to the new tool. The goal is to enable the robot to generalize its task knowledge to accommodate tool replacements and thus be more robust to changes in its environment. We demonstrate how the tool transform models learned from one episode of task corrections can be used to perform that task with $\geq$ 85% of maximum performance in 83% of tool/task combinations. Furthermore, these transformations generalize to unseen tool/task combinations in 27.8% of our transfer evaluations, and up to 41% of transfer problems when the source and replacement tool share tooltip similarities. Overall, these results indicate that successful task adaptation for a new tool is dependent on the the tool's usage within that task, and that the transform model learned from interactive corrections can be generalized to other tasks providing a similar context for the new tool.

KEYWORDS

Human-robot interaction; Interactive learning; Transfer learning

ACM Reference Format:
Tesca Fitzgerald, Elaine Short, Ashok Goel, and Andrea Thomaz. 2019. Human-guided Trajectory Adaptation for Tool Transfer. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Tools are common in human life, and thus a robot that operates in human environments is likely to encounter situations in which it needs to use tools as well. While a robot can easily learn to complete a new task with a new tool via demonstrations by a human teacher, the demonstration(s) provided for that tool cannot prepare the robot for all variations of that tool it is likely to encounter. These variations can range from different tool dimensions (e.g. different sized spoons, hammers, and screwdrivers) to tool replacements when a typical tool is not available (e.g. using a measuring cup instead of a ladle, or a rock instead of a hammer).

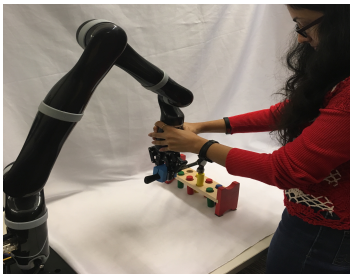


Figure 1: The robot receiving a task demonstration

Although some tools are flexible, many have a rigid structure that results in the tool having two properties that can be used to improve transfer: (1) that a task is performed with respect to a particular acting surface of that tool (which we refer to as the tooltip), and (2) that there is a fixed relationship between the robot's gripper and the tooltip. By learning the relationship between the robot's gripper trajectories when performing the same task with two different tools, our approach aims to learn the relationship between the tools themselves and how they are used in the context of that task. Once this relationship is learned, the robot can apply it to reuse previously-learned task models with the new tool.

While the relationship between the robot's gripper and the tooltip is static, modeling the relationship between the tooltip and the robot's motion is challenging. In general, one would expect to be able to solve this problem by modifying the trajectory so that the tooltip of the new tool followed the same path as the tooltip of the original tool. However, there are several key challenges to this approach. First, the problem of identifying the tooltip is non-trivial due to ambiguities, such as how any point of a cup's rim may be the tooltip for a pouring action. Second, a different tooltip may be used depending on the task, such as how the rim of a ladle is used for scooping, while the back of the scoop can be used to push or pull another object away. As a result, the transfer function learned for one class of tasks (e.g. pushing or scooping) may not be reusable for all other tasks completed with the same tool. Additionally, some parts of a task are under-constrained (such as the robot moving a measuring cup toward a bowl), whereas other parts of the same task are constrained with respect to the tooltip (such as the edge of the measuring cup when pouring it).

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

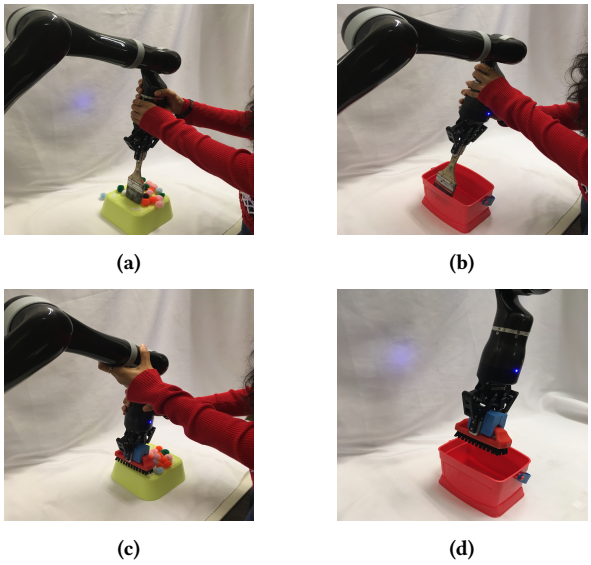


Figure 2: The robot receives demonstrations of sweeping (a) and hooking (b) tasks using the first tool (a paintbrush). After receiving corrections of the sweeping task (c) using a new tool (a short scrub-brush), the robot uses these corrections to complete an undemonstrated tool-task combination: hooking the box with the scrub-brush (d).

In this paper, we introduce an algorithm for *transfer by correction*: an interactive approach to learning the relationship between tools such that tasks learned using one tool (such as in Fig. 2a) can quickly be transferred to utilize a new tool (Fig. 2c). The robot interacts with a human teacher to receive corrections when repeating a known task with a new tool, pausing to enable the teacher to correct the position and orientation of its gripper throughout the task (e.g. to correct a collision, or correct its location with respect to a target object). The algorithm then represents these corrections according to two *tool transform models* in order to identify the pose transformation that is most consistent across corrections.

Our results indicate that the tool transform models learned from one episode of task corrections can be used effectively to model the relationship between the source and replacement tool, enabling it to **achieve ≥ 85% of maximum performance in 83% of tool/task combinations**. Furthermore, we test the generalizability of the learned transformation to additional tasks (such as in Fig. 2d), and find that the tool transform model improves transfer performance in 27.8% of across-task evaluations, and 41% of across-task evaluations in which the source and replacement tool share similarities in their tooltips. Overall, our work demonstrates that (i) we can effectively model the transforms between tools using interactive corrections, and (ii) the transform can be generalized to other tasks providing a similar context for the new tool, **without additional corrections, nor any training on those tool-task combinations**.

2 BACKGROUND

Tool Manipulation: A robot situated in human environments will encounter environments and tasks suited for human capabilities, and thus it is important for a robot to be able to use human tools [17]. The shape of a tool alters its effect on its environment [23], and thus a tool replacement may necessitate a change in the manipulation of that tool in order to achieve the same task goal [6]. For tasks involving the use of a rigid tool, the static relationship between the robot's hand and the tooltip is sufficient for controlling the tool to complete a task [15, 16]. These methods assume a single tooltip for each tool, and that this tooltip is detected via visual or tactile means. For tasks involving multiple surfaces of the tool, the task model can be explicitly defined with respect to those segments of the tool and repeated with tools consisting of similar segments [14]. However, this assumes a hand-defined model that represents the task with respect to pre-defined object segments, and that these object segments are shared across tools. Given enough training examples of a task, a robot can learn a success classifier that can later be used to self-supervise learning task-oriented tool grasps and manipulation policies for unseen tools [9]. We similarly aim to situate a new tool in the context of a known task, but eliminate the assumptions that (i) the new tool is within the scope of the training examples (which would exclude atypical tool replacements) and (ii) that the tool features relevant to the task are observable and recorded by the robot.

Task Learning and Corrections: An advantage of Learning from Demonstration is that the robot can quickly receive new demonstrations [2, 7] for a new tool variation by having the teacher physically guide it to repeat the task with the new tool. A more efficient approach, however, would be for the robot to learn about the relationships between tools, such that this relationship can be extended to transfer multiple task models to be reused with the new tool. Due to the unstructured nature of task demonstrations, the original and new demonstrations may vary in ways that do not reflect accommodations necessary to repeat the task using the new tool. Interactive *corrections* have been shown to be an effective interface for adapting a previously-learned task model [3, 4, 21]. We leverage this form of interaction for tool transfer. In doing so, we minimize the distance between the original and corrected goal poses throughout the task, thus increasing the likelihood that these corrections reflect *only* the trajectory changes *necessary* for the new tool.

Transfer and One-shot Learning: The aim of transfer learning for reinforcement learning domains is typically to use feedback obtained during exploration of a new environment in order to enable reuse of a previously learned model [25]. In previous work, we have shown how interaction can be used to transfer the high-level ordering of task steps to a series of new objects in a target domain [12]. Similarly, the aim of one-shot learning is to quickly learn a new task, often improving learning from a single demonstration by adapting previous task knowledge. Prior work in this space focuses on learning a latent space for the task in order to account for new robot dynamics [24] or new task dynamics [13, 18]. "Meta-learning" approaches have succeeded at reusing visuomotor task policies learned from one demonstration [10] and using a new goal state to condition a learned task network such that it can be reused with additional task objects [8]. We address the problem of a robot that has *not yet* been able to explore these relationships, aiming to enable rapid adaptation of a task model for unseen task/parameter relationships. The tool transform models learned by our approach are not specific to any task learning algorithm or representation,

and thus can compliment or bootstrap methods for reinforcement, one-shot, and meta learning.

3 PROBLEM DEFINITION

Suppose that for a particular task, there exists a transfer function ϕ_a^b that transforms pose $\mathbf{P_a}$ using tool a into a pose $\mathbf{P_b}$ for tool b:

$$\mathbf{P_b} = \phi_a^b(\mathbf{P_a}) \tag{1}$$

We assume that each demonstration consists of several keyframes [1]. The robot receives corrections by executing a trajectory planned using the original task model, pausing after a time interval defined by the keyframe timings set during the original demonstration. The teacher then moves the robot's gripper to the correct position, after which the robot resumes task execution for the next time interval, repeating the correction process until the entire task is complete. Each resulting correction at interval i consists of the original pose $\mathbf{C_a^i}$ (using tool a) and the corrected pose $\mathbf{C_b^i}$ (using new tool b) at keyframe i. A collection of K corrections (one for each of K keyframes) results in a K x 2 correction matrix:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C_a^0} & \mathbf{C_b^0} \\ \mathbf{C_a^1} & \mathbf{C_b^1} \\ \cdots & \\ \mathbf{C_a^K} & \mathbf{C_b^K} \end{bmatrix} \tag{2}$$

Each corrected pose $\mathbf{C_b^i}$ provides a sample of the transfer function value with the original pose $\mathbf{C_a^i}$ at keyframe i as input, plus some amount of error from the optimal correction pose:

$$\mathbf{C_b^i} = \phi_a^b(\mathbf{C_a^i}) + \epsilon \qquad \epsilon_n \sim \mathcal{N}(0, \sigma_n^2) \tag{3}$$

We assume ϵ is sampled from a Gaussian noise model for each axis $n \in [1 \ldots 6]$ of the 6D end-effector pose. Our aim is to learn a transfer function ϕ that optimally reflects the task constraints, using a correction matrix $\mathbf{C}$. Our research questions are as follows:

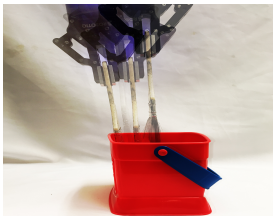
(1) How can we learn ϕ for a particular task from $\mathbf{C}$ containing sparse, noisy corrections?
(2) Under what conditions can the ϕ learned from corrections on one task be used to transfer other known tasks to the same replacement tool? What characteristics of the tool and task predict whether a previously-learned ϕ can be applied?

4 APPROACH: TRANSFER BY CORRECTION

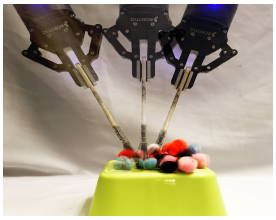
Given a task trajectory $\mathbf{T}$ for tool a consisting of a series of t poses in task space such that $\mathbf{T} = [\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_t}]$, we transform each pose individually for tool b. Representing an original pose for tool a in terms of its 3 x 1 translational vector $\mathbf{t_a}$ and 4 x 1 rotational vector $\mathbf{r_a}$, we transform it into a pose $\mathbf{p_b}$ for tool b as follows:

$$\mathbf{p_b} = \phi_a^b(\mathbf{p_a}) = \left\langle \mathbf{t_a} + \hat{\mathbf{t}}, \mathbf{r_a} \cdot \hat{\mathbf{r}} \right\rangle \tag{4}$$

Here, $\mathbf{r_a} \cdot \hat{\mathbf{r}}$ refers to the Hamilton product between the two quaternions. The goal is now to estimate the optimal rotational $\hat{\mathbf{r}}$ and translational $\hat{\mathbf{t}}$ transformation components from the corrections matrix $\mathbf{C}$, and then apply these transformations to the trajectory $\mathbf{T}$. Our approach addresses this goal by (1) modeling $\mathbf{C}$, particularly the relationship between each correction's translational and rotational components, (2) sampling a typical translational transformation $\hat{\mathbf{t}}$ and rotational transformation $\hat{\mathbf{r}}$ from this transform model, and (3) applying $\hat{\mathbf{t}}$ and $\hat{\mathbf{r}}$ to transform each pose in the task trajectory according to Eq. 4.



(a) Orientation constraint



(b) Tooltip constraint

Figure 3: Poses meeting the same orientation constraint share similar orientations but vary more in their position, whereas poses meeting the same tooltip constraint rotate around the tooltip.

4.1 Task Constraints

We observe that corrections indicate constraints of the tooltip's position and/or orientation, and that these constraints are reflected in the relationship between the translation and rotation components of each correction. Broadly, each correction may primarily indicate:

- An *unconstrained* point in the trajectory, and thus should be omitted from the tool transform model.
- An *orientation constraint*, where the rotation of the tooltip (and thus the end effector) is constrained more than its position (e.g. hooking a box is constrained more by the orientation of the hook than its position, as in Fig 3a).
- A *tooltip constraint*, where the position of the tooltip is constrained more than its rotation (e.g. sweeping a surface with a brush). Note that the *tooltip* position is the center of this constraint rather than the end-effector itself, and thus the range of valid end-effector positions forms an arc around the tooltip, and its orientation remains angled toward the tooltip (e.g. Fig 3b).

We define two *tool transform models*, each reflecting either orientation or tooltip constraints. We fit the corrections matrix to each tool transform model, using RANSAC [11] to iteratively estimate the parameters of each model while discarding outlier and unconstrained correction data points. Each iteration involves (i) fitting parameter values to a sample of n datapoints, (ii) identifying a set of inlier points that also fit those model parameters within an error bound of ϵ, and (iii) storing the parameter values if the inlier set represents a ratio of the dataset $> d$. The RANSAC algorithm relies on a method for fitting parameters to the sample data, and a distance metric for a datapoint based on the model parameters. These are not defined by the RANSAC algorithm, and so we specify the parameterization and distance metric according to the tool transform model used, which we describe more in the following sections. We define an additional method to convert the best-fitting parameters following RANSAC completion into a typical transform that can be applied to poses.

4.2 Linear Tool Transform Model

Based on the *orientation* constraint type, we first consider a linear model for correction data, where corrections fitting this model

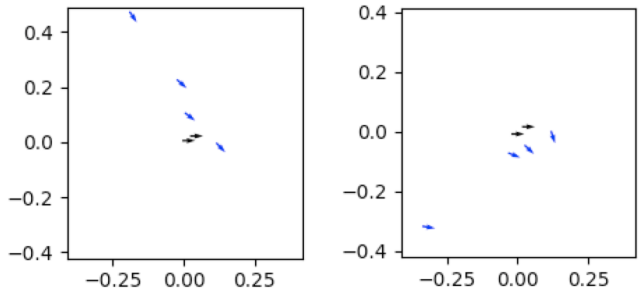


Figure 4: Each plot represents one set of corrections for a task. The position of each arrow represents the change in $< x, y >$ position, and points in the direction of the change in orientation introduced by that correction. Orientation constraints can be seen in (a), where the majority of corrections on this tool have low variance in their orientation, but higher variance in their x-y position. Tooltip constraints can be seen in (b), where the majority of corrections arc around a singular center of rotation, and orientation is dependent on the x-y position. Unconstrained keyframes (colored grey) are located near (0,0).

share a linear relationship between the translational components of the corrections, while maintaining a constant relationship between the rotational components of corrections (visualized in Fig. 4a). We model this linear relationship as a series of coefficients obtained by applying PCA to reduce the 3D position corrections to a 1D space.

4.2.1 RANSAC Algorithm Parameters. The RANSAC algorithm is performed for k iterations, where we use the estimation

$$k = \frac{log(1.0 - p)}{log(1.0 - w^n))} \tag{5}$$

with desired confidence $p = 0.99$ and estimated inlier ratio $w = 0.5$. Additional parameters are as follows: $n = 2$ is the number of data points sampled at each RANSAC iteration, $\epsilon = 0.01$ is the error threshold used to determine whether a data point fits the model, and $d = 0.5$ is the minimum ratio between inlier and outlier data points in order for the model to be retained.

4.2.2 Model Parameter Fitting. Model fitting during each iteration of RANSAC consists of reducing the datapoints to a 1D model using PCA, returning the mean translational correction and the coefficients for the first principal component of the sample $\mathbf{S}$:

$$\Theta_{\text{linear}}(\mathbf{S}) = \left\langle \theta_\mu, \theta_\mathbf{u} \right\rangle \qquad \theta_\mu = \frac{1}{|\mathbf{S}|} \sum_{\mathbf{p} \in \mathbf{S}} \mathbf{p_t} \tag{6}$$

where $\mathbf{p_t}$ is the 3 x 1 translational difference indicated by the correction $\mathbf{p}$, $\mathbf{S}$ is the subset of the corrections matrix $\mathbf{C}$ sampled during one iteration of RANSAC such that $\mathbf{S} \subset \mathbf{C}$, and $\theta_\mathbf{u}$ is the eigenvector corresponding to the largest eigenvalue of the covariance matrix $\Sigma = \frac{1}{|\mathbf{S}|} \mathbf{S_t}^T \mathbf{S_t}$.

4.2.3 Error Function. Each iteration of RANSAC calculates the total error over all data points fitting that iteration's model parameters. We define the error of a single correction datapoint $\mathbf{p}$ as the sum of its reconstruction error and difference from the average orientation correction, given the current model parameters θ:

$$\begin{aligned} \delta_{\text{linear}}(\mathbf{p}, \theta) =& \|\mathbf{p_t} - (\theta_\mu + (\mathbf{p_t} - \theta_\mu)^T \theta_\mathbf{u} {\theta_\mathbf{u}}^{T^+})\| \\ &+ \gamma \left((1 - \overline{\mathbf{q}}_\mathbf{n} {\mathbf{p_n}}^T)^2 \right) \end{aligned} \tag{7}$$

where $\mathbf{x}^+$ indicates the Moore-Penrose pseudo-inverse of a vector, $\mathbf{p_n}$ is the unit vector representing the orientation difference indicated by the correction $\mathbf{p}$, $\overline{\mathbf{q}}_\mathbf{n}$ is a unit vector in the direction of the average rotation sampled from the model (defined in the next section), and γ is the weight assigned to rotational error ($\gamma = 1$ in our evaluations).

4.2.4 Sampling Function. After RANSAC returns the optimal model parameters and corresponding set of inlier points $\hat{\mathbf{I}} \subset \mathbf{C}$, the rotation and translation components of the transformation are sampled from the model. We define the sampling function according to the estimated "average" rotation $\overline{\mathbf{q}}$:

$$\Psi(\hat{\mathbf{I}}, \hat{\theta})_{\text{linear}} = \left\langle \overline{\mathbf{q}}, \overline{\mathbf{t}} \right\rangle \tag{8}$$

$$\overline{\mathbf{q}} = \arg\max_{\mathbf{q} \in \mathbb{S}^3} \mathbf{q}^T M \mathbf{q} \qquad M = \frac{1}{|\hat{\mathbf{I}}|} \sum_{\mathbf{p} \in \hat{\mathbf{I}}} {\mathbf{p_q}}^i {\mathbf{p_q}}^{i^T} \tag{9}$$

The solution to $\overline{\mathbf{q}}$ for this maximization problem is the eigenvector corresponding to the largest eigenvalue of M [19]. The sample translation $\overline{\mathbf{t}}$ is the 3D offset corresponding to the mean value $\overline{z}$ from the 1D projection space:

$$\overline{\mathbf{t}} = \hat{\theta}_\mu + \overline{z} \hat{\theta_\mathbf{u}}^{T^+} \qquad \overline{z} = \frac{1}{|\hat{\mathbf{I}}|} \sum_{p \in \hat{\mathbf{I}}} (\mathbf{p}_t - \hat{\theta}_\mu)^T \hat{\theta}_\mathbf{u} \tag{10}$$

4.3 Rotational Tool Transform Model

We now consider a model for corrections reflecting a *tooltip constraint*, in which we make the assumption that corrections indicate a constraint over the tool tip's *position*. Since the tool tip is offset from the end-effector, the position and rotation of the end-effector are constrained by each other such that the end-effector revolves around the tool tip (visualized in Fig. 4b). We model this relationship by identifying a center-of-rotation (and corresponding rotation radius) for the tool tip, from which we can sample a valid end-effector position and rotation.

4.3.1 RANSAC Algorithm Parameters. We use the same parameters for k, w, d as in the linear model. We sample $n = 3$ points at each iteration, and use the error threshold $\epsilon = 0.25$. We define functions for model parameterization, error metrics, sampling, and variance in the following sections.

4.3.2 Model Parameter Fitting. We define the optimal model parameters for each iteration of RANSAC as the center-of-rotation (and corresponding rotation radius) of that iteration's samples $\mathbf{S}$:

$$\Theta_{\text{rotation}}(\mathbf{S}) = \langle \theta_\mathbf{c}, \theta_r \rangle \tag{11}$$

where $\theta_\mathbf{c}$ is the position of the center-of-rotation that minimizes its distance from the intersection of lines produced from the position and orientation of each correction sample:

$$\theta_\mathbf{c} = \arg\min_\mathbf{c} \sum_{i=1}^{|S|} D(\mathbf{c}; \mathbf{a_i}, \mathbf{n_i})^2 \tag{12}$$

where $\mathbf{a_i}$ and $\mathbf{n_i}$ are the position and unit direction vectors, respectively, for sample i in S:

$$\mathbf{a}_i = [x_i, y_i, z_i]^T \qquad \mathbf{n}_i = \left(\mathbf{q}_i \cdot [0,1,0,0]^T)\right) \cdot \mathbf{q}' \tag{13}$$

Here, $\mathbf{q_1} \cdot \mathbf{q_2}$ refers to the Hamilton product between two quaternions, and $\mathbf{q}'$ is the inverse of the quaternion $\mathbf{q}$:

$$\mathbf{q}' = [w, x, y, z]'^T = [w, -x, -y, -z]^T \tag{14}$$

We solve for the center-of-rotation by adapting a method for identifying the least-squares intersection of lines [26]. We consider each sample i to be a ray originating at the point $\mathbf{a_i}$ and pointing in the direction of $\mathbf{n_i}$. The center-of-rotation of a set of these rays is thus the point that minimizes the distance between itself and each ray. We define this distance as the piecewise function:

$$D(\mathbf{c}; \mathbf{a}, \mathbf{n}) = \begin{cases} \|(\mathbf{c} - \mathbf{a}) - d \cdot \mathbf{n}\|_2 & \text{if } d > 0 \\ \|\mathbf{c} - \mathbf{a}\|_2 & \text{otherwise} \end{cases} \tag{15}$$

where d is the distance between $\mathbf{a}$ and the projection of the candidate centerpoint $\mathbf{c}$ on the ray:

$$d = (\mathbf{c} - \mathbf{a})^T \mathbf{n} \tag{16}$$

We solve for $\theta_\mathbf{c}$ using the SciPy implementation of the Levenberg-Marquardt method for non-linear least-squares optimization, supplying Eq. 15 as the cost function. We then solve for the radius corresponding to $\theta_\mathbf{c}$:

$$\theta_r = \frac{1}{|\mathbf{S}|} \sum_{i=0}^{|S|} \|\mathbf{a_i} - \theta_\mathbf{c}\| \tag{17}$$

4.3.3 Error Function. We define the error of a single data point $\mathbf{p}$ as its distance from the current iteration's center-of-rotation estimate:

$$\delta_{\text{rotation}}(\mathbf{p}, \theta) = \left(\frac{D(\mathbf{c}; \mathbf{a_p}, \mathbf{n_p})}{d_p}\right)^2 \tag{18}$$

where d_p is defined in Eq. 16.

4.3.4 Sampling Function. After RANSAC returns the optimal model parameters and corresponding set of inlier points $\hat{\mathbf{I}} \subset \mathbf{C}$, the rotation component of the transformation is first sampled using the "average" rotation $\overline{\mathbf{q}}_c$ from $\hat{\theta_\mathbf{c}}$ to all inlier points:

$$\overline{\mathbf{q}}_c = \arg\max_{\mathbf{q} \in \mathbb{S}^3} \mathbf{q}^T M \mathbf{q} \qquad M = \frac{1}{|\hat{\mathbf{I}}|} \sum_{\mathbf{p} \in \hat{\mathbf{I}}} \mathbf{r_p} \mathbf{r_p}^T \tag{19}$$

where $\mathbf{r_p}$ is the quaternion rotation between $\hat{\theta_\mathbf{c}}$ and the position of $\mathbf{p}$, defined by normalizing the quaternion consisting of the scalar and vector parts:

$$\mathbf{r_p} = \left\langle \|\mathbf{a}\|^2 + \mathbf{b}\mathbf{a}^T, \mathbf{b}^T \times \mathbf{a} \right\rangle \tag{20}$$

$$\mathbf{a} = \mathbf{p}_t - \hat{\theta_\mathbf{c}} \qquad \mathbf{b} = [\|\mathbf{a}\|, 0, 0] \tag{21}$$

The optimal $\overline{\mathbf{q}}_\mathbf{c}$ is the eigenvector corresponding to the largest eigenvalue of M; this represents the sampled rotation from $\hat{\theta_\mathbf{c}}$.

We then sample $\bar{\mathbf{t}}$ by projecting the point at distance $\hat{\theta}_r$ from $\hat{\theta_\mathbf{c}}$ in the direction of $\overline{\mathbf{q}}_\mathbf{c}$:

$$\bar{\mathbf{t}} = \hat{\theta_\mathbf{c}} + \left[(\overline{\mathbf{q}}_\mathbf{c} \cdot [0, \hat{\theta}_r, 0, 0]^T) \cdot \overline{\mathbf{q}}_\mathbf{c}'\right]_{1..3} \tag{22}$$

where $\mathbf{x}_{1..3}$ indicates the 3 x 1 vector obtained by ommitting the first element of a 4 x 1 vector $\mathbf{x}$. Finally, we return the sample consisting of the translation $\bar{\mathbf{t}}$ and the normalized rotation $\overline{\mathbf{q}}$ between $\bar{\mathbf{t}}$ and $\hat{\theta_\mathbf{c}}$:

$$\Psi(\hat{\mathbf{I}}, \hat{\theta})_{\text{rotation}} = \left\langle \frac{\overline{\mathbf{q}}}{\|\overline{\mathbf{q}}\|}, \bar{\mathbf{t}} \right\rangle \tag{23}$$

$$\overline{\mathbf{q}} = \left\langle \hat{\theta}_r \|\mathbf{a}\| + \mathbf{b}\mathbf{a}^T, \mathbf{b}^T \times \mathbf{a} \right\rangle \tag{24}$$

$$\mathbf{a} = \hat{\theta_\mathbf{c}} - \bar{\mathbf{t}} \qquad \mathbf{b} = [\hat{\theta}_r, 0, 0] \tag{25}$$

4.4 Best-Fit Model Selection

The linear and rotational tool transform models represent two different relationships between the translational and rotational components of corrections. We now define a metric for selecting between these two models based on how well they fit the correction data:

$$\Psi(\mathbf{C})_{\text{best-fit}} = \begin{cases} \Psi(\hat{\mathbf{I}}_l, \hat{\theta}_l)_{linear} & \text{if } \Delta_{linear} < \Delta_{rotation} \\ \Psi(\hat{\mathbf{I}}_r, \hat{\theta}_r)_{\text{rotation}} & \text{otherwise} \end{cases} \tag{26}$$

where $\hat{\mathbf{I}}_l, \hat{\theta}_l, \hat{\mathbf{I}}_r, \hat{\theta}_r$ represent the optimal inlier points and parameter values from the linear and rotational models, respectively. The fit of the linear model is calculated as its range of values $\mathbf{z}$ projected in the model's 1D space:

$$\Delta_{linear} = \text{range}(\mathbf{z}) \qquad \mathbf{z} = \{(\mathbf{p}_t - \hat{\theta}_\mu)^T \hat{\theta}_\mathbf{u} | \forall \mathbf{p} \in \hat{\mathbf{I}}\} \tag{27}$$

The fit of the rotational model is calculated as the range of unit vectors in the direction of each inlier point as measured from the center-of-rotation:

$$\Delta_{rotation} = 1 - \frac{1}{|\hat{\mathbf{I}}|} \left\| \sum_{p \in \hat{\mathbf{I}}} \left[(\mathbf{r_p} \cdot [0,1,0,0]^T) \cdot \mathbf{r_p}'\right]_{1..3} \right\|_2 \tag{28}$$

where $\mathbf{r_p}$ is defined in Eq. 20.

5 EVALUATION

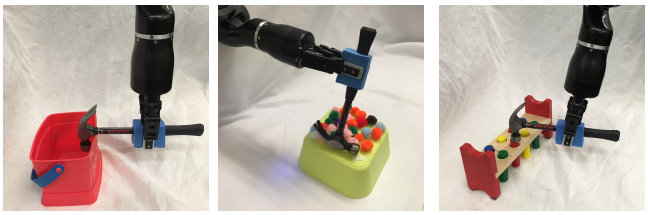
We evaluated the transfer by correction algorithm results on a 7-DOF Jaco2 arm equipped with a Robotiq 85 gripper and mounted vertically on a table-top surface (pictured in Fig. 1). Each evaluation configuration consisted of: (i) one task demonstration provided using the *source* tool, (ii) the new, *replacement* tool, and (iii) one correction task (demonstrated with the source tool, and used to obtain corrections with the replacement tool). We describe data collection for each of these steps in the following sections.

5.1 Demonstrations

Three tasks (Fig. 6) were demonstrated using three prototypical, "source" tools (Fig. 5a-c), resulting in a total of 9 demonstrations. Demonstrations began with the arm positioned in an initial configuration and with the gripper already grasping the tool. Objects on the robot's workspace were reset to the same initial position before every demonstration. We provided demonstrations by indicating keyframes [1] along the trajectory, each of which was reached by moving the robot's arm to the intermediate pose. At each keyframe, the 7D end effector pose was recorded; note that this is the pose of the joint holding the tool, and *not* the pose of the tool-tip itself (since the tool-tip is unknown to the robot). We provided one keyframe demonstration for each combination of tasks

(a) (b) (c) (d) (e)

Figure 5: Tools a-c were used to demonstrate the three tasks shown in Fig. 6, later transferred to use tools d-e. These tools exhibit a wide range of grasps, orientations, dimensions, and tooltip surfaces.



(a) (b) (c)

Figure 6: (a) Hooking task, (b) sweeping task, and (c) hammering task

and source tools in this manner, each demonstration consisting of 7-12 keyframes (depending on the source tool used) for the sweeping task, 10-11 keyframes (depending on the source tool used) for the hooking task, and 7 keyframes for the hammering task. Following each demonstration, a Dynamic Movement Primitive (DMP) model [20, 22] was trained on the recorded keyframe trajectory. DMPs represent a demonstration as a stable dynamical system and are generalizable to variations in start and end pose constraints. We re-recorded the demonstration if the trained DMP failed to repeat the demonstration task with the source tool.

5.2 Corrections

Following training, the arm was reset to its initial configuration, with the gripper already grasping a new tool (Fig. 6d-e). Objects on the robot's workspace were reset to the same initial position as in the demonstrations. The learned model was then used to plan a trajectory in task-space, which was then converted into a joint-space trajectory using TracIK [5] and executed, pausing at intervals defined by the keyframe timing used in the original demonstration. When execution was paused, it remained paused until the arm pose was confirmed. If no correction was necessary, the pose was confirmed immediately; otherwise, the arm pose was first corrected by moving the arm to the correct position.

Two poses were recorded for each correction: (i) the original end-effector pose the arm attempted to reach (regardless of whether the goal pose was reachable with the new tool), and (ii) the end-effector pose following confirmation (regardless of whether a correction was given). Trajectory execution then resumed from the arm's current pose, following the original task-space trajectory so that pose corrections were not propagated to the rest of the trajectory. This process continued until all keyframes were corrected and executed, resulting in the correction matrix $\mathbf{C}$ (Eq. 2).

5.3 MEASURES

For each transfer execution, we measured performance according to a metric specific to the task:

- *Sweeping:* The number of pom-poms swept off the surface of the yellow box.
- *Hooking:* The final distance between the box's target position and the closest edge of the box (measured in centimeters).
- *Hammering:* A binary metric of whether the peg was pressed any lower from its initial position.

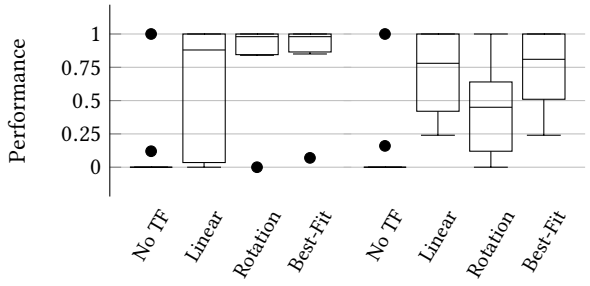
5.4 RESULTS

We highlight two categories of results: *within*-task and *across*-task performance.

5.4.1 Within-task Transfer. *Within-task* performance measures the algorithm's ability to model the corrections and perform the corrected task successfully. Transfer was performed using the transform model learned from corrections on *that same task/tool combination*. For example, for the sweeping task model learned using the hammer, corrections were provided on the replacement tool (e.g. a mug) and then used to perform the sweeping task using that same mug. For each source tool, we evaluated performance on all 3 tasks using each of the 2 replacement objects, resulting in 18 sets of corrections (one for each combination of task, source tool, and replacement tool) per tool transform model (linear and rotational).

We scaled the result of each transfer execution between 0 and 1, with 0 representing the initial state of the task and 1 representing maximum performance according to the metrics in Sec. 5.3. Using the better-performing model resulted in $\geq 85\%$ of maximum task performance in 83% of cases. The better-performing model was selected using the best-fit metric in 72% of cases. Fig. 8 lists the percentage of transfer executions (using the best-fit model) that achieve multiple performance thresholds, where best-fit results were recorded as the performance of the model returned by Eq. 26.

Figure 7 reports the performance distribution aggregated over all tasks, transferred from each of the 3 source tools to either the scrub-brush (pictured in Fig. 5e, results in Fig. 7a) or mug (pictured in Fig. 5d, results in Fig. 7b) as the replacement tool. The mean performance results are reported in Fig. 10a, with darker cells indicating better performance. Overall, the transform returned using the best-fit metric resulted in average performance of 6.9x and 5.9x that of the untransformed trajectory when using the scrub-brush and mug, respectively, as replacement tools.

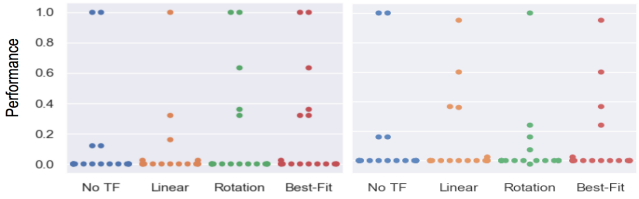


(a) Brush (b) Mug

Figure 7: Results for within-task transfer using the scrub-brush or mug as the replacement tool. Performance was measured according to the metrics in Sec. 5.3, scaled between 0-1.

Performance Threshold	95%	85%	75%	65%	55%	45%
Transfer Executions	61%	72%	72%	72%	83%	89%
Untransformed Executions	11%	11%	11%	11%	11%	11%

Figure 8: Percentage of within-task transfer executions (selected by best-fit model) and untransformed trajectories achieving various performance thresholds (defined as the % of maximum performance metric for that task, described in Sec. 5.3)



(a) Brush (b) Mug

Figure 9: Results for across-task transfer using the scrub-brush or mug as the replacement tool. Performance was measured according to the metrics in Sec. 5.3, scaled between 0-1.

5.4.2 Across-task Transfer. *Across-task* transfer performance measures the generalizability of corrections learned on one task when applied to a *different* task using the same tool, without having received any corrections on that tool/task combination. For example, the hooking task was learned using the hammer, and transferred to the mug using corrections obtained on the sweeping task. We evaluated 36 total transfer executions (one per combination of demonstration task, source tool, correction task (distinct from the demonstration task), and replacement tool) per tool transform model (linear and rotational).

Figure 9 reports the performance distribution aggregated over all tasks, transferred from each of the 3 source tools to either the scrub-brush (Fig. 9a) or mug (Fig. 9b) as the replacement tool. The mean performance results are reported in Fig. 10b, with darker cells indicating better performance. Overall, the transform returned using the best-fit metric resulted in average performance of 1.6x and 0.94x that of the untransformed trajectory when using the scrub-brush and mug, respectively, as replacement tools. The performance distribution is improved when using the transform learned from corrections, resulting in 2.25x as many task executions achieving $\geq$ 25% of maximum task performance.

In order to understand the conditions under which a transform can be reused successfully in the context of another task, we also report the mean performance results for a subset of the across-task executions (Fig. 10c). This subset consists of only the task executions where the relative orientation is the same between (i) the source tool's tooltips used for the source and target tasks and (ii) the replacement tool's tooltips used for the same two tasks. This subset consisted of 10 executions for the scrub-brush, and 12 for the mug. Overall, for this subset of executions, the transform returned using the best-fit metric resulted in average performance of 12.6x and 1.7x that of the untransformed trajectory when using the scrub-brush and mug, respectively, as replacement tools.

Replacement Tool	No TF	Linear	Rotation	Best-fit
Brush	0.124	0.648	0.850	**0.863**
Mug	0.129	0.738	0.489	**0.765**

(a) Mean performance of within-task transfer to the brush and mug replacement tools over all 18 transfer executions for each tool.

Replacement Tool	No TF	Linear	Rotation	Best-fit
Brush	0.124	0.085	0.184	**0.203**
Mug	**0.129**	0.128	0.080	0.121

(b) Mean performance of across-task transfer to the brush and mug replacement tools over all 18 transfer executions for each tool.

Replacement Tool	No TF	Linear	Rotation	Best-fit
Brush	0.024	0.053	0.268	**0.302**
Mug	0.097	0.162	0.101	**0.162**

(c) Mean performance of across-task transfer to the brush and mug replacement tools over the subset of transfer executions in which the transformation between source and correction tasks is similar for the source and replacement tool (10 executions for the brush, 12 for the mug).

Figure 10: Mean performance for within-task, across-task, and a subset of across-task transfer executions. Darker cells indicate higher average performance.

6 DISCUSSION

Our within-task transfer evaluation tested whether we can model the transform between two tools in the context of the same task (represented by the solid blue arrow in Figure 11) using corrections. Our results indicate that one round of corrections is sufficient to indicate this relationship between tools; collectively, the linear and rotational models achieved $\geq$ 85% of maximum task performance in 83% of cases. Individually, the models selected by the best-fit metric

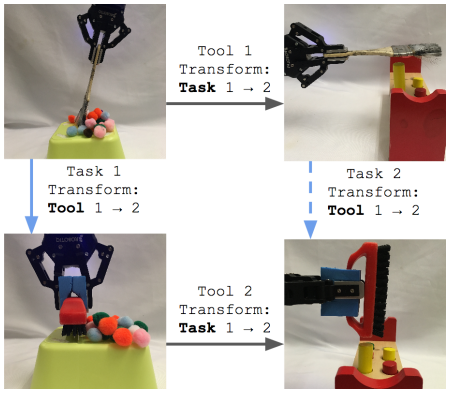


Figure 11: Corrections indicate the transform from tool 1 to tool 2 for the same task (indicated by the solid blue arrow). Our within-task transfer evaluation tested whether we can use corrections to sufficiently model this relationship. Different tasks may use different tooltips from the same tool (such as the different tooltips used to complete tasks 1 and 2). Our across-task evaluation tests whether the transform learned from corrections (solid blue arrow) can be reused as the transform between the two tools for another task (indicated by the dashed blue arrow).

achieved this performance threshold in 72% of cases. This indicates that, in general, the fit of the model itself can be used to indicate the relationship between end-effector position and orientation for a given tool/task combination.

The primary benefit of modeling corrections (as opposed to re-learning the task for the new tool) is two-fold: First, the robot learns a transformation that reflects *how* the task has changed in response to the new tool, which is potentially generalizable to other tasks (as we discuss next). We hypothesize that in future work, this learned transform could be parameterized by features of the tool (after corrections on multiple tools). Second, since we do not change the underlying task model, but instead apply the learned transform to the resulting trajectory, the underlying task model is left unchanged. We expect that this efficiency benefit would be most evident when transferring a more complex task model trained over many demonstrations; rather than require more demonstrations with the new tool in order to re-train the task model, the transform would be applied to the result of the already-trained model.

We have also explored how well this transform generalizes to other tasks. Different tooltips on the same tool may be used to achieve different tasks, such as how the end and base of the paintbrush are used to perform sweeping and hammering tasks, respectively, in Fig. 11. While we do not explicitly model the relationship between tooltips on the same tool (represented by the top grey arrow in Fig. 11), they are inherent to the learned task models. A similar relationship exists for the replacement tool (represented by the bottom grey arrow in Fig. 11). Our across-task evaluation seeks to answer whether the relationship between tools in the context of the first task (solid blue arrow) can be reused for a second task (represented by the dashed blue arrow) without having received any corrections on that tool/task combination (tool 2 and task 2). While we see lower performance in across-task evaluations compared to the within-task evaluations, it does improve transfer in 27.8% of across-task transfer executions (in comparison to the untransformed trajectory).

In the general case, our results also indicate that we cannot necessarily reuse the learned transformation on additional tasks, as average performance in across-task transfer is slightly worse than that of the untransformed trajectory when the mug is used as a replacement tool. This presents the question: given a transform between two tools in the context of one task, under what conditions can that transform be reused in the context of another task *without additional corrections or training*? We do see that across-task performance is greatest when considering only the subset of cases where the relationship between the tooltips used in either task is similar for the source and replacement tools (in our evaluation, this is 10 of 18 executions using the brush, and 12 of 18 executions using the mug). Within this subset, across-task transfer improves performance in 41% of transfer executions. From this we draw two conclusions: (i) the transform applied to a tool is contextually dependent on the source task, target task, and tooltips of the source and replacement tool, and (ii) a transform can be reused when the relationship between tooltips used in either task is similar for the source and replacement tools.

7 CONCLUSION

We have presented a method for transfer by correction: repeating a known task with an unknown tool in order to record a human teacher's corrections of the robot's motion. We have contributed two models for representing corrections, a linear and rotation model, that each represent a different relationship between the end-effector's position and orientation when using a tool. We have also presented a metric for choosing the better-fitting model for a set of corrections.

In our within-task evaluation, we have demonstrated that either the linear or rotational model is sufficient to represent corrections for successful task completion with the new tool in 83% of task executions. Furthermore, using a metric to select the best-fitting model resulted in improved performance in 89% of tasks (in comparison to the original, untransformed trajectory).

Our across-task evaluation tests the generalizability of the transforms learned from corrections to additional tasks, *without any additional training or corrections*. We observed that across-task transfer improved performance in 27.8% of task executions, and that further improvement is seen in transfer scenarios where the relationship between the tooltips used on the source tool is similar to that of the replacement tool. Overall, these results indicate that successful task adaptation for a new tool is dependent on the the tool's usage within that task, and that the transform model learned from interactive corrections can be generalized to other tasks providing a similar context for the new tool.

ACKNOWLEDGMENTS

We thank Dr. Sonia Chernova for many helpful discussions in planning the evaluation. This material is based on work supported by ONR grant N000141410120 and the IBM PhD Fellowship.

REFERENCES

[1] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. 2012. Keyframe-based learning from demonstration. *International Journal of Social Robotics* 4, 4 (2012), 343–355.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57, 5 (2009), 469–483.

[3] Brenna D Argall, Eric L Sauser, and Aude G Billard. 2010. Tactile guidance for policy refinement and reuse. In *IEEE 9th International Conference on Development and Learning (ICDL)*. IEEE, 7–12.

[4] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. 2018. Learning from Physical Human Corrections, One Feature at a Time. In *ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 141–149.

[5] Patrick Beeson and Barrett Ames. 2015. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 928–935.

[6] Solly Brown and Claude Sammut. 2012. Tool use and learning in robots. In *Encyclopedia of the Sciences of Learning*. Springer, 3327–3330.

[7] Sonia Chernova and Andrea L Thomaz. 2014. Robot Learning from Human Teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8, 3 (2014), 1–121.

[8] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-shot imitation learning. In *Advances in Neural Information Processing Systems*. 1087–1098.

[9] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. 2018. Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision. In *Robotics: Science and Systems*. Pittsburgh, Pennsylvania. https://doi.org/10.15607/RSS.2018.XIV.012

[10] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. 2017. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905* (2017).

[11] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[12] Tesca Fitzgerald, Ashok Goel, and Andrea Thomaz. 2018. Human-Guided Object Mapping for Task Transfer. *ACM Trans. Hum.-Robot Interact.* 7, 2, Article 17 (Oct. 2018), 24 pages. https://doi.org/10.1145/3277905

[13] Justin Fu, Sergey Levine, and Pieter Abbeel. 2016. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4019–4026.

[14] Pawel Gajewski, Paulo Ferreira, Georg Bartels, Chaozheng Wang, Frank Guerin, Bipin Indurkhya, Michael Beetz, and Bartlomiej Sniezynski. 2018. Adapting Everyday Manipulation Skills to Varied Scenarios. *arXiv preprint arXiv:1803.02743* (2018).

[15] Heiko Hoffmann, Zhichao Chen, Darren Earl, Derek Mitchell, Behnam Salemi, and Jivko Sinapov. 2014. Adaptive robotic tool use under variable grasps. *Robotics and Autonomous Systems* 62, 6 (2014), 833–846.

[16] Charles C Kemp and Aaron Edsinger. 2006. Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *International Conference on Development and Learning (ICDL)*, Vol. 42.

[17] Charles C Kemp, Aaron Edsinger, and Eduardo Torres-Jara. 2007. Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics & Automation Magazine* 14, 1 (2007), 20–29.

[18] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. 2017. Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes. In *Advances in Neural Information Processing Systems*. 6250–6261.

[19] F Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. 2007. Averaging quaternions. *Journal of Guidance, Control, and Dynamics* 30, 4 (2007), 1193–1197.

[20] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. 2009. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 763–768.

[21] Eric L Sauser, Brenna D Argall, Giorgio Metta, and Aude G Billard. 2012. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems* 60, 1 (2012), 55–71.

[22] Stefan Schaal. 2006. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*. Springer, 261–280.

[23] Jivko Sinapov and Alexadner Stoytchev. 2008. Detecting the functional similarities between tools using a hierarchical representation of outcomes. In *IEEE International Conference on Development and Learning (ICDL)*. IEEE, 91–96.

[24] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. 2018. Universal Planning Networks. *arXiv preprint arXiv:1804.00645* (2018).

[25] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research* 10 (2009), 1633–1685.

[26] J Traa. 2013. Least-squares Intersection of Lines. *UIUC, Illinois* (2013). http://cal.cs.illinois.edu/~johannes/research/LS_line_intersect.pdf