Shape-Based Transfer of Generic Skills

Skye Thompson MIT CSAIL
rsthomp@mit.edu

Leslie Pack Kaelbling MIT CSAIL
lpk@csail.mit.edu

Tomas Lozano-Perez MIT CSAIL
tlp@csail.mit.edu

***Abstract*— We propose a new, data-efficient approach for skill transfer to novel objects, accounting for known categorical shape variation. A low-dimensional shape representation embedding is learned from a set of deformations, sampled between known objects within a category. This latent representation is mapped to a set of control parameters that result in successful execution of a category-level skill on that object. This method generalizes a learned manipulation policy to unseen objects with few training examples. We demonstrate this approach on pouring from cups and scooping with spatulas, where there is complex, nonlinear variation of successful control parameters across objects.**

I. Introduction

A primary concern of research in robotic manipulation is generalization in representation: How can a robot learn, from a small number of example demonstrations of a skill, to apply that same skill to many differently-shaped objects? We focus on a broad class of skills in which particular aspects of object geometry dictate the details of skill execution.

The objects used in many manipulation skills have similar geometric features that afford that manipulation, and define a natural category: for example, we might expect the objects used in a scooping task all to have some holding surface on the end of a handle. A control parameterization for a skill can leverage these affordances to generalize across a category. The key to effective generalization is to identify the shape features that play a functional role in the object's use. By definition, those are exactly the features that are in common across all objects in the category: the class of pitchers may have a wide variety of aesthetic shape variability that is irrelevant to pouring, but the features they do share are those that are critical to their function.

In this paper, we describe a strategy that combines unsupervised learning of a latent shape description of a functional category of objects with learning of skills, leveraging that latent space, from a one or two expert demonstrations and a small number of self-supervised experiments. We compare it to existing methods based directly on non-rigid registration (illustrated in Figure 1) on problems of pouring from cups and scooping with spatulas, and demonstrate that it can learn, from even a single expert demonstration, manipulation strategies that are substantially more robust to natural category variability.

We gratefully acknowledge support from NSF grant 1723381; from AFOSR grant FA9550-17-1-0165; from ONR grant N00014-18-1-2847; from the Honda Research Institute, from MIT-IBM Watson Lab; and from SUTD Temasek Laboratories.

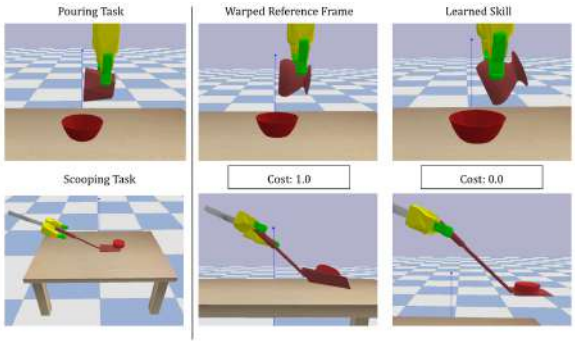

Fig. 1: We demonstrate our method on a pouring and scooping task. The first frame in each row shows a demonstration on a canonical instance of the shape class. The second frames show that a state-of-the-art *warping* of that demonstration completely fails to solve the problem for a novel shape. The third frames show results of our method, leveraging a learned shape embedding to transfer the demonstration to novel objects.

II. Problem Definition

The problem of *shape-based skill transfer* (SST) is to learn a skill from a small number of examples, in cases where the detailed execution of the skill is driven by the shape of a single object that is being operated on or used as a tool. The shape of those objects can vary, but they are all contained in a category in which the parameters of the required control strategy can be expressed with a continuous dependence on the object's shape. We describe the approach in terms of a single object, though it could be extended directly to operations that involve multiple objects, using a similar mechanism to make the control depend on the shapes of all of the objects.

The performance task is: given a new object, characterized by its shape $s \in S$, specified by a set of M 3D points given as a $M \times 3$ matrix, to generate trajectory parameters that will perform a desired skill effectively, where the success of a particular skill execution on a particular object can be measured by a scalar *cost* in $\mathbb{R}$. We assume a fixed-dimensional parameter space $\Psi = \mathbb{R}^d$ that specifies a trajectory of object poses, and a low-level robot controller that can achieve that pose sequence with linear Cartesian interpolation. We ultimately seek a policy $\pi : S \rightarrow \Psi$ that maps shapes to trajectory parameters, in a way minimizing expected cost over the distribution of object shapes found in the world.

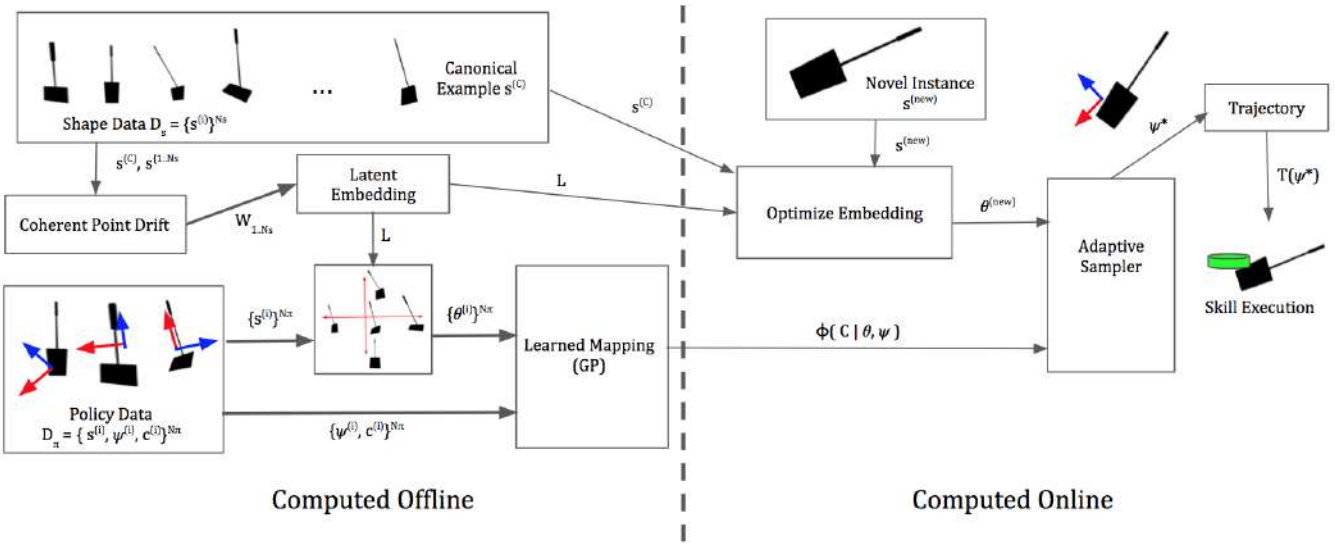

Fig. 2: Illustration of the method, including offline learning phase and online adaptation to a new object.

We are given data in two forms:

- Shape data $\mathcal{D}_S = \{s^{(i)}\}_{i=1}^{N_S}$, where $s^{(i)} \in S$ for all i. This data characterizes the distribution of shapes that the learned skill needs to cover.
- Policy data $\mathcal{D}_\pi = \{(s^{(i)}, \psi^{(i)}, c^{(i)})\}_{i=1}^{N_\pi}$ where $s^{(i)} \in S$, $\psi^{(i)} \in \Psi$, and $c^{(i)} \in \mathbb{R}$. This data indicates, for combinations of shapes and control parameters, the cost of executing that control trajectory with that shape.

We separate these datasets because, when learning, it may be easier to get multiple demonstrations using the same few objects, and be difficult to get demonstrations using many different objects. however, object *shapes* from the category might be easy to acquire via another channel, such as a collection of object images, point-clouds, or meshes. Our goal is to be able to learn the skill policy efficiently from a small number of examples in each of these data sets.

III. Approach

Our approach is to decompose the policy learning into two learning sub-problems. We define the policy

$$\pi(s) = \arg\min_{\psi \in \Psi} C(E(s), \psi) \ ,$$

where E is a learned embedding of S into a low-dimensional latent space $\Theta = \mathbb{R}^d$, and $C : \Theta \times \Psi \to \mathbb{R}$ is a function mapping a shape representation $E(s)$ and control parameters ψ into the cost of executing the trajectory resulting from parameters ψ on an object with shape s in the world. Because of the structure of the provided data, it is natural to use the shape data to learn an embedding representation for shapes, and the policy data to learn the cost function given that embedding. One counterproposal might be to train the system "end-to-end" rather than performing an unsupervised representation-learning phase, but we would not be most effectively using both of our data sets in that mode.

We take the approach of learning a cost function on pairs of a shape and controller parameters, rather than the more direct strategy of learning a direct mapping from shape to control parameters for several reasons:

- There may be many suitable, but very different, control parameters ψ for a single s, and we do not want to average them during training. For example, we could successfully flip a pancake over from the left or from the right, but the average of those strategies would fail.
- It is very difficult to get the expert-only control strategies that would be needed to train the direct policy; by learning the cost function, we can make use of non-expert demonstrations as well as the results of the robot's own experimentation, resulting in more training data and more robust learning.
- Only a subset of possible ψ values can be feasibly executed by the robot; constraining a regression model to only produce valid outputs can be difficult.
- We may want to use this learned skill in a larger context, such as a task and motion planning system [1], [2], where some variations of the control (such as flipping the pancake from the left) might be impossible in the current situation because of environmental constraints (such as occlusions by other objects or kinematic constraints of the robot) not directly considered when learning the controller. Knowing a scoring function enables selecting diverse candidate actions by intelligent sampling of the set of controls that have high scores for the current object shape under some given precondition.

A. Learning the shape embedding

It is notoriously difficult to represent 3D shapes as points in a vector space in a way that enables effective generalization. We follow the strategy, described by Rodriguez and Behnke [3], which rather than representing the shape of each object directly, represents *the difference*, articulated as a continuous warping transformation, between each object and a canonical object from the set. These transformations are

simpler to describe and, hence, easier to learn, than general shape models.

1) Selecting the canonical instance: We begin by finding a canonical instance of the given category; it will be an element of the shape data set $\mathcal{D}_S = \{s^{(i)}\}_{i=1}^{N_S}$. We assume that elements s of the set S of shapes are represented by M points sampled uniformly from the volume contained in the object's watertight mesh (note that the interior of a cup is *not* included in this sampling).

Our objective will be to represent the shapes of all other objects $s^{(i)}$ in the category as some transformation of the canonical object $s^{(C)}$, so that $s^{(i)} \approx \mathcal{T}_i(s^{(C)})$ for all i. It is important to note that this means that the shape $s^{(i)}$ will be approximately the same as the shape $\mathcal{T}_i(s^{(C)})$, but the order of points in the vector will not necessarily be the same.

The ideal canonical instance will have the property that it can be accurately transformed to most other instances, but this metric is difficult to evaluate for objects with many complex features. Instead, we have found that a simple distance metric is easy to optimize and results in reliable selection of effective canonical shapes. Our metric on shapes is the summed Cartesian distances between each point in one shape $s^{(i)}$ and the closest point on shape $s^{(j)}$:

$$D(s^{(i)}, s^{(j)}) = \sum_{m=1}^{M} \min_n \|s_m^{(i)} - s_n^{(j)}\| \ .$$

It is important to note that this metric is *insensitive* to the ordering of the individual points in the shape descriptions within the matrices $s^{(i)}$ and $s^{(j)}$, so there is no requirement for them to "match up." We select the canonical object to be the one that has the minimum summed distance to all other objects:

$$s^{(C)} = \arg\min_{s \in \mathcal{D}_S} \sum_{i=0}^{N_S} D(s, s^{(i)}) \ .$$

Informal experimentation has shown that this strategy for selecting a canonical object performs well in practice.

2) Transforming the canonical instance: We model the transformation between the canonical object and instance $s^{(i)}$, $\mathcal{T}_i$, as an additive offset to each point in the canonical object,

$$\mathcal{T}_i(s^{(C)}) = s^{(C)} + W_i \ ,$$

where W_i is an $M \times 3$ matrix. Although simply letting $W_i = s^{(i)} - s^{(C)}$ would result in perfect reconstruction of the input shapes, it would be highly sensitive to the ordering of the points in the representation of $s^{(i)}$; this would make the W_i for different examples shapes be incommensurate and would not serve as an effective representation for future learning.

Instead, again following Rodriguez and Behnke [3], we use the *coherent point drift* (CPD) algorithm proposed by Myronenko and Song [4]. CPD is an approach to non-rigid registration that models the points in target instance $s^{(i)}$ as samples from a Gaussian mixture model (GMM) with component means that are the transformed points in $s^{(C)}$, with the mixture weights determined by the distance between a point and the means of the mixture components and with additional regularization constraining neighboring points to move in similar ways.

CPD finds matrix W_i by minimizing the negative log likelihood of all of the 3D points in the vector representing the transformed version of $s^{(i)}$, which is $s^{(C)} + W_i$, assuming they are drawn according to the GMM specified by $s^{(C)}$. This leads us to a regularized cost function

$$J(W_i) = -\sum_{n=1}^{M} \log \sum_{m=1}^{M} \exp \left(-\frac{1}{2\sigma^2} \|s_n^{(i)} - (s^{(C)} + W_i)_m\| \right) + \frac{\lambda}{2}\phi(W_i) \ ,$$

where $\phi(W)$ is a regularization of the displacement matrix, and λ represents a trade-off between regularization and goodness-of-fit. Crucially, this likelihood measure is *insensitive* to the ordering of the points in the description of $s^{(i)}$: each point in the transformed canonical shape has to be near *some* point in $s^{(i)}$, but the elements W_i are "ordered" according to the ordering of points in $s^{(C)}$, so the W_i can be sensibly seen as living continuously in the same space. The objective $J(W_i)$ is minimized using *expectation maximization*, yielding a warp matrix W_i for each training shape $s^{(i)}$.

3) Constructing the latent space of transformations: In order to represent shape variation effectively for computing the warp transforms, the number of points representing an object, M, needs to be high enough to model relevant object features. In our experiments, each cup was represented by approximately 1500 points, and each spatula by approximately 1000. Our goal is to learn to map a new shape to effective control parameters from few examples; to achieve such a low sample complexity, we need a much more compact representation of shapes. So, we seek a low-dimensional embedding of the warp matrices W, much smaller than the complete pointcloud representation.

We begin by "flattening" each $M \times 3$ warp matrix W_i into a length $3M$ vector $\overline{W_i}$, and let $\overline{W}$ be the stack of N_S vectors $\overline{W_i}$. A straightforward strategy for dimensionality reduction is *principal components analysis* (PCA). We perform PCA on $\overline{W}$ to reduce the dimensionality of our transforms from $3M$ to D, obtaining a $3M \times D$ matrix L such that $W_i L$ is now a low-dimensional representation of shape $s^{(i)}$.

An alternative strategy for finding a low-dimensional embedding of the shape warps would be to use a neural-network auto-encoder. We experimented with this method but found that PCA was more effective, particularly in the reduced-data regime. The table below shows mean reconstruction error for two methods of finding a low-dimensional embedding: PCA and an auto-encoder with a RELU-activation hidden layer. Both were trained on warps between a canonical spatula shape and between 6 and 200 other spatula examples, restarted 5 times. The metric shown is the average reconstruction error per point, in meters, when using the 5-dimensional latent space representation produced by each model to reconstruct 10 randomly selected novel spatulas. The auto-encoder's performance degrades more quickly as

the number of training examples is reduced, making PCA the preferred option for small numbers of example objects.

examples	6	50	100	500
PCA	0.0465	0.0396	0.0407	0.0262
	$\pm$0.0014	$\pm$0.0030	$\pm$0.0048	$\pm$0.0006
auto-encoder	0.0504	0.0404	0.0448	0.0296
	$\pm$0.0065	$\pm$0.0148	$\pm$0.0101	$\pm$0.0070

B. Learning the cost function

Now that we have constructed a compact representation of shapes, we can focus on learning the cost function. Our first step is to transform the policy training data $\mathcal{D}_\pi = \{(s^{(i)}, \psi^{(i)}, c^{(i)})\}_{i=1}^{N_\pi}$ using the shape representation we have just constructed, as shown in the left of Figure 2. Letting D be the dimension used in PCA, each shape $s^{(i)}$ can be represented as an element $\theta^{(i)} \in \mathbb{R}^D$ where $\theta^{(i)} = L\,W_i$, allowing us to re-represent our data as

$$\mathcal{D}_\pi = \{(\theta^{(i)}, \psi^{(i)}, c^{(i)})\}_{i=1}^{N_\pi} \ .$$

Now all components are real-valued vectors or scalars suitable for standard ML methods.

We train a *Gaussian process* (GP) regression model on this data, which yields a distribution $\phi(C \mid \theta, \psi)$ on the cost of executing control ψ on an object with shape θ. A GP is an effective model for relatively low-dimensional data, compared to pixel or pointcloud representations, and provides an estimate of certainty along with its output prediction, allowing us to select for minimum cost predictions with higher confidence. This is a useful strategy when selecting control parameters, as it allows us to identify the "safest" proposed parameters for a given shape that are unlikely to result in unpredictable behavior.

C. Making predictions

When the robot encounters a new object with shape $s^{(\text{new})}$ to use in performing the learned skill, we begin by finding a representation for it in Θ space. If we had a complete point-cloud for the new shape, then we might take the direct approach of using CPD to find the transform W_{new} from $s^{(C)}$ to $s^{(\text{new})}$ and then letting $\theta^{(\text{new})} = L\,W_{\text{new}}$.

However, while we assumed that we had full point-cloud models for shapes at training time, when we need to perform a skill with a novel object, we may only have access to a partial or occluded model. So, in order to get a more robust representation of the new object in Θ space, we search in the latent space for a representation that is a good match, allowing us to find a best fit for even incomplete models.

A point $\theta^{(o)} \in \Theta$ can be post-multipied by L^T and reshaped to obtain a transformation W_o that transforms S^C to a candidate point set $s^{(o)}$. Our aim is to find the $\theta^{(o)}$ that produces an $s^{(o)}$ closest to $s^{(\text{new})}$, as measured by minimizing the summed closest-point distance D. So, we would like to find

$$\theta^{(\text{new})} = \arg\min_\theta (D(s^{(\text{new})}, s^{(C)} + \theta\,L^T)) \ .$$

We find $\theta^{(\text{new})}$ by performing gradient descent in the latent space, beginning at a randomly sampled initial proposal point θ_0. This strategy is much more robust in the face of partial shape descriptor $s^{(\text{new})}$.

Now that we have found a latent-space representation of the novel object $\theta^{(\text{new})}$, we randomly sample a set of proposal control parameters $\psi_{1..N}$ from the range of possible values and select the $\psi^{(\text{new})}$ with the lowest predicted upper bound on C:

$$\psi^{(\text{new})} = \arg\min_{\psi \in \psi_{1..N}} (\mu_C(\psi) + .25\sigma_C(\psi)) \ .$$

For sufficiently low-dimensional Ψ, this is a feasible way to identify the best control parameters. For a higher-dimensional Ψ, it may be necessary to select potential $\psi^{(\text{new})}$ to evaluate using a more informed approach, such as gradient descent, to identify $\psi^{(\text{new})}$. Success by this metric means a low cost when executing the task on a novel object. The selected parameters $\psi^{(\text{new})}$ are used to generate a trajectory for execution, $T(\psi^{(\text{new})})$.

IV. Related Work

There are a number of existing approaches to transfer manipulation skills across objects varying in shape.

Schulman et al. [5] and Lee et al. [6] describe a method for adapting task demonstrations across objects of varying geometry by performing non-rigid registration between a known scene and a novel one. The warp obtained between these scenes is then applied to a demonstrated trajectory found to be successful in the known scene. This approach is limited to skills where the warping transformations can be applied to both shape and trajectory parameters. Furthermore, it may not be the case that the transformation between shapes will be effective for the trajectory for some skills. This is illustrated in Figure 1. Even for these simple skills, the transformed trajectories are not successful, because the impact of the novel objects' geometry on a successful trajectory is not adequately accounted for.

Manuelli et al. [7] describe a rigid keypoint representation to generalize a manipulation skill across a category. They train a deep neural network to identify a set of handpicked, labeled keypoints that are useful for defining a task on images of objects in a category. They then plan a path between the keypoints' initial and desired final positions to complete the task. Florence and Manuelli [8] describe a method of grasp transfer using dense visual object descriptors to establish functional equivalency across deformed objects. Our method, in comparison, learns features from geometric models and requires less training data.

Hillenbrand and Roa [9] present a method for grasp transfer relying on local replanning around a warped contact point. Rodriguez and Behnke [3] explore the transfer of grasping skills to novel objects by learning a linear latent-space representation of object features. Sampled grasps on a range of objects are transformed to form a distribution of grasps on the canonical object, and a linear regressor is trained to predict each grasp given the latent-space representation. Our method relies on a similar latent object

representation, but instead of transforming our control policy into the canonical object's space and learning a regression, or locally replanning, we instead learn the nonlinear cost function, choosing the control parameters conditioned on the object's representation. This allows us to learn a more general class of skills, and also supports sampling multiple solutions.

Brandi et al. [10] capture intra-category variation by warping between objects, tracking the warping of a selected set of measured features, which are then used as parameters in a probabilistic motion primitive (PROMP) that describes the skill policy. When executing the skill on a novel object, a trajectory is sampled from the PROMP, and transformed by the measured warps of selected features. Our approach instead discovers important features through learning a low-dimensional embedding, rather than explicitly measuring important selected features. We also learn the potentially nonlinear relationship between trajectory and execution cost, rather than constrained variations of the trajectory based on measured features.

V. EXPERIMENTS

We demonstrate our SST method on two tasks and compare its performance to a state-of-the-art baseline as well as several ablations.

A. Task domains

We consider two fairly different task domains, implemented in a PyBullet physics simulation: pouring small particles out of cups of varying shapes and scooping up a "patty" with spatulas of varying shapes and lengths.

a) Control parameterization: In both skills, the control space ψ is 6-dimensional, $(x_1, y_1, z_1, x_2, y_2, z_2)$, where (x_1, y_1, z_1) represent a point in 3-space and the vector between that point and (x_2, y_2, z_2) represent its orientation. We choose this representation because it is effective and because it simplifies comparison with the baseline method, which requires a point-based representation. We can think of ψ as an oriented point on the object, although in fact it need not actually be physically on the object's surface.

The pouring skill has a fixed target frame f, which is centered on the target receptacle in x and y, and 10cm above its top in z. The controller, $T(\psi)$, interpolates between the initial pose of object frame ψ and an object pose specified by $\psi = f$. This interpolation is linear but happens in two separate phases: first the cup is translated so the positions of ψ and f are equal. Then the cup is rotated to match the orientations.

The scooping skill has a fixed target frame f, centered on the patty in x, 25 cm forwards in y, and 40 cm above it in z. This controller also has two stages: first, it interpolates between the initial pose of frame ψ on the spatula and an intermediary pose aligned with the $X - Y$ position of f, with Z-axis perpendicular to the table surface, over 25 time steps; then it moves ψ between the intermediary pose and the target frame f, linearly interpolating all 6 axes over 25 time steps.

b) Cost functions: The cost of a pouring skill execution is determined by the fraction of the 40 particles that begin inside the cup that do not end up in the target receptacle. The cost of a scooping skill execution is the distance of the patty from its final goal position on top of the elevated spatula.

c) Data sets and experiment procedure: Figure 3 illustrates the objects used in both tasks and shows the effectiveness of the basic WARP strategy on each one.

For each skill, we begin with one or two expert demonstrations on each object in $\mathcal{D}_\pi$. We use the GP-UCB algorithm [11] to select additional values $\psi^{(k)}$, for each distinct $\theta^{(i)}$, where it would be most informative to carry out additional self-supervised experiments, and compute the associated cost values by executing them in the PyBullet simulation. The results are added to the dataset and the process is repeated.

In the pouring experiments, we use a set of 12 cup shapes, each represented as a set of meshes comprising a convex decomposition of the shape (the convex decomposition is necessary for PyBullet to simulate containment of the particles.) We generate shape data-sets $\mathcal{D}_S$ by taking random subsets of 7 of the 12 cups. We generate policy data-sets $\mathcal{D}_\pi$ by collecting a set of 50 training executions with each cup, driven by GP-UCB as described above. The control parameters selected by GP-UCB are passed to the simulator to produce a cost score. To illustrate the capacity of SST to learn multi-modal skills, we initialize the GP with two good demonstration pours, one from the left and one from the right, so the dataset will include both types of examples.

The setup is identical for the scooping experiments, except that only a single demonstration is used to initialize the GP for each spatula, scooping from the left.

In both domains, to execute an experiment, we (1) select a random subset of 7 objects for the shape data set, (2) select 4 of the remaining objects for the policy data set, train the SST, and then evaluate its performance by executing the skill in simulation 20 times on the held-out object and recording its cost. Finally, for both tasks, we test with partial point-cloud observations of the objects at performance time.

B. Comparisons

We compare SST against several baselines: warping, and two ablations of SST.

- **Warping** (W) directly warps a demonstration from a known object to a novel one using the learned warp matrix W_i, by identifying the points representing the reference frame ψ for the demonstration pour on the previously identified canonical object, calculating the warp matrix W between it and the novel object, and applying this transformation to the points representing the demonstration reference frame, resulting in the reference frame of the novel object. The demonstration points are selected from the points of the canonical object so that their new location after the application of the warp W can be extracted.
- **Features** (FSST) is a method that lets the shape representation Θ be a set of ideal handpicked features

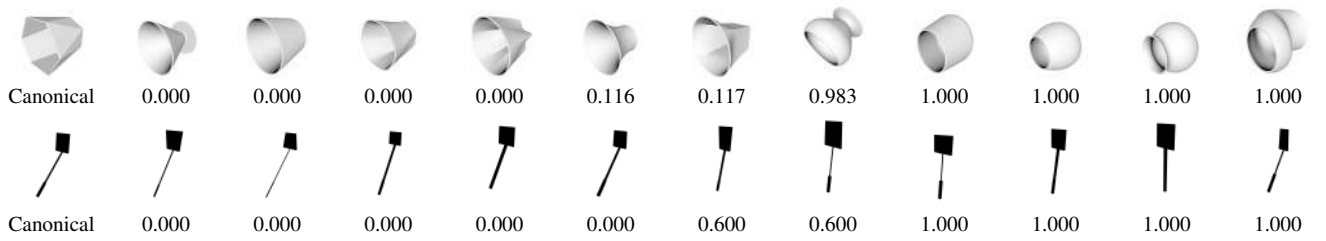

Fig. 3: Objects used in the experiments; numbers indicate cost of WARPing a demonstration on the first object in the row to the others.

rather than a learned latent space, but uses SST's second phase unchanged. We include this to represent an "upper bound" on performance and measure the effectiveness of the latent-space learning. For cups, these parameters are the height, maximum diameter, lip diameter, and height of the widest point. For spatulas, they are the angle of the handle, the thickness of the spatula head, the length and width of the head, and angle of the tip.

- **Regress** (R) is a method that learns a regression directly from θ. In the case of cups, one of the two demonstration pours, from the right or from the left, is randomly selected. A GP is trained on the 5 datapoints, predicting the control parameters ψ from the object's shape.

C. Results

We perform this process, beginning with the random selection of objects to form the latent feature space, 5 times, performing 20 actions with each of the 5 objects excluded for testing, evaluating 500 actions total for each method. The difference between the cost of each action and the averaged cost of the warping baseline performed on each provided demonstration are reported below, where a negative value indicates lower cost (better performance) than the baseline.

The table below contains the results of these experiments. We report the mean scores relative to score of the warp method and perform paired statistical analysis in that space to decrease variance, reporting 95% confidence intervals on the means. We observe that SST performs significantly better than WARP on both experiments. We ran SST on both complete and partial point clouds and the performance was indistinguishable. We note also that SST with ideal features instead of the learned latent space (FSST, in the table) works roughly as well as SST, which demonstrates that the latent-space learning is very effective. Finally, we see that the ablation of SST in which we perform regression from shape parameters directly to controls performs very poorly in both cases, having particular difficulty with the fact that there are two distinct modes for pouring, which highlights the importance of our strategy of learning a scoring function.

	Pour	Scoop
SST-W	-0.295 ± 0.046	-0.362 ± 0.044
FSST-W	-0.286 ± 0.048	-0.393 ± 0.046
R-W	$+0.531 \pm 0.194$	$+0.272 \pm 0.296$

Our method outperforms the warping of control parameters between spatulas and regression between the shape and control parameters, and performs comparably to the use of handpicked measurements.

VI. Discussion and Future Work

Usage of the information available in the geometry of a manipuland's shape produces useful features that allow the robot to learn to successfully execute a task. By estimating the variation in shape within a category, we can create a useful embedding representation that is robust to categorical variations in shape without hand-engineering. This representation allows us to achieve efficient generalization to new objects within a category, requiring fewer training objects to manipulate, and fewer executions on the novel object to achieve good results. This approach generalizes to a novel object more efficiently than the use of objects' geometric features with no consideration for known intra-category variation in learning, as it requires less training data and time. It generalizes more effectively than warping the initial reference frame from one example object to new objects without learning, as it better captures nonlinear relationships between object shape and control parameters. Our formulation of this problem also allows for consideration of additional contextual parameters beyond object geometry during skill learning. For the purpose of this work, we focus solely on object geometry, but the inclusion of additional environmental information as features at the skill learning stage would be a straightforward extension of our method.

In future work, we could extend the method to require weaker initial models, working from partial point-clouds of the training shapes. This might also allow the robot to take advantage of cross-category similarities, by identifying object features (for example, the "cup" part of a mug, eliminating the handle) and producing a representation for those known parts. We focused on quasi-static skills in this paper, which can be learned without explicitly modeling inertial properties like weight or friction. Tasks requiring these properties could be addressed in the future with this method through the use of relevant control parameters.

In summary, finding the shape commonalities across a functional category of objects and constructing a representation of those commonalities forms the basis of a method for learning to score skill executions effectively from a small amount of data, learning to generalize to novel objects.

REFERENCES

[1] Z. Wang, C. Garrett, L. Kaelbling, and T. Lozano-Perez, "Active model learning and diverse action sampling for task and motion planning," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4107–4114, 2018.

[2] Z. Wang, C. R. Garrett, L. Kaelbling, and T. Lozano-Perez, "Learning compositional models of robot skills for task and motion planning," *ArXiv*, vol. abs/2006.06444, 2020.

[3] D. Rodriguez and S. Behnke, "Transferring Category-Based Functional Grasping Skills by Latent Space Non-Rigid Registration," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2662–2669, 2018.

[4] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2262–2275, 2010.

[5] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," *Springer Tracts in Advanced Robotics*, vol. 114, pp. 339–354, 2016.

[6] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 177–184, 2015.

[7] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kPAM: KeyPoint Affordances for Category-Level Robotic Manipulation," *ArXiv*, 2019. [Online]. Available: http://arxiv.org/abs/1903.06684

[8] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation," *ArXiv*, pp. 1–12, 2018. [Online]. Available: http://arxiv.org/abs/1806.08756

[9] U. Hillenbrand and M. A. Roa, "Transferring functional grasps through contact warping and local replanning," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2963–2970, 2012.

[10] S. Brandi, O. Kroemer, and J. Peters, "Generalizing pouring actions between objects using warped parameters," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Febru, pp. 616–621, 2015.

[11] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, 2002.