ToolFlowNet: Robotic Manipulation with Tools via Predicting Tool Flow from Point Clouds

Daniel Seita, Yufei Wang[†]**, Sarthak J. Shetty**[†]**, Edward Yao Li**[†]**, Zackory Erickson, David Held**
[†]Equal contribution.
The Robotics Institute, Carnegie Mellon University, USA
Correspondence to: `dseita@andrew.cmu.edu`

Abstract: Point clouds are a widely available and canonical data modality which convey the 3D geometry of a scene. Despite significant progress in classification and segmentation from point clouds, policy learning from such a modality remains challenging, and most prior works in imitation learning focus on learning policies from images or state information. In this paper, we propose a novel framework for learning policies from point clouds for robotic manipulation with tools. We use a novel neural network, ToolFlowNet, which predicts dense per-point flow on the tool that the robot controls, and then uses the flow to derive the transformation that the robot should execute. We apply this framework to imitation learning of challenging deformable object manipulation tasks with continuous movement of tools, including scooping and pouring, and demonstrate significantly improved performance over baselines which do not use flow. We perform 50 physical scooping experiments with ToolFlowNet and attain 82% scooping success. See `https://tinyurl.com/toolflownet` for supplementary material.

Keywords: Flow, Point Clouds, Tool Manipulation, Deformables

1 Introduction

Recently, learning-based techniques have become effective for improving the generalization capabilities of robots for manipulation tasks such as grasping [1], reorienting [2], rearrangement [3], and tossing [4]. Data observations tend to be either images [5, 6, 7] or state information such as joint angles or end-effector poses [8], which are passed into a deep network to obtain an output vector encoding an action, typically representing a change in end-effector pose or joint angles. While these approaches have shown a wide range of successes, a fundamental limitation has to do with the nature of the observation. Using images requires projecting information into a 2D space which might lose valuable 3D information. Furthermore, learning from images in simulation often leads to a sim2real gap [9] in performance. Although it is easy to access the internal robot states such as joint angles, the robot does not necessarily have the ground-truth state of objects in the environment, which might require complex state estimation systems [10]. Moreover, it is hard to define a state for deformable objects like liquid and cloth [11, 12].

In this work, we propose a framework for learning robotic manipulation from point cloud observations. Point clouds are a canonical data modality and are widely available from camera sensors, providing valuable information about the structure of the 3D space [13, 14]. However, policy learning from point clouds has been less explored compared to learning from images or state, potentially owing to the difficulty of reasoning about raw 3D point cloud inputs. While there have been many proposed architectures which are specialized for learning from point clouds [13, 15, 14, 16, 17], these works tend to focus on computer vision tasks such as classification and segmentation. Policy learning from point clouds, while feasible in some contexts [18, 19], remains challenging.

We study learning from point clouds for robotic manipulation tasks with tools. The input data is a segmented point cloud which, for each point, contains its 3D coordinates and a one-hot vector

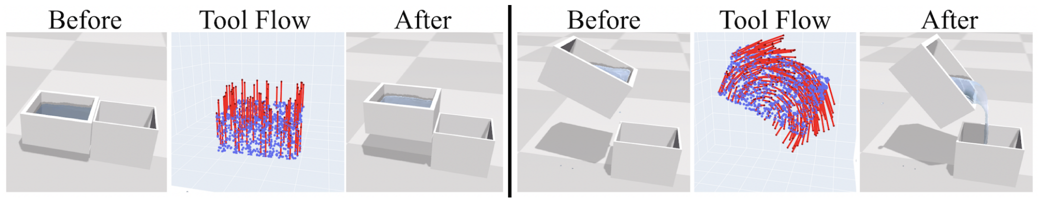

Figure 1: ToolFlowNet applied on a pouring task in simulation, where the tool is the box which contains water. Given a point cloud (colored blue), ToolFlowNet learns dense per-point flow vectors (colored red), which describe the intended 3D motion of each tool point. These are converted to translation and rotation actions. Left: the tool moves upwards. Right: the tool rotates to pour water. We subsample the flow for visual clarity.

indicating the object class the point belongs to. Our key insight is to use *dense representations* and *flow* to represent the tool action. We build upon dense point-cloud processing architectures [15] and train the model to predict per-point output values which we call *tool flow*. This represents the 3D movement of each tool point in a point cloud from one time step to the next, which is an instance of scene flow [20]. Our model is trained with Behavioral Cloning on tool flow data, which provides a dense per-point supervision. Given the set of tool flow vectors, we convert flow to an SE(3) transformation, which represents the actual action a robot would execute. We call this model ToolFlowNet and visualize it in Figure 1 for a pouring task in simulation. We compare this against non-dense methods which directly regress to an action and demonstrate the benefits of tool flow as an action representation. To summarize:

- We propose a general framework for learning from segmented point clouds for manipulation with tools by utilizing a novel architecture, ToolFlowNet, which predicts per-point tool flow vectors.
- We show how to train ToolFlowNet for imitation learning and explore different loss functions for training. We perform extensive ablation studies to validate these choices.
- We perform simulated imitation learning experiments on scooping and pouring tasks and show the benefits of using ToolFlowNet over baselines which do not use flow.
- We demonstrate ToolFlowNet achieves 82% success rate on 50 physical scooping trials.

2 Related Work

Point Clouds and Flow Researchers have proposed a variety of architectures specialized for learning from point clouds [13, 15, 14, 21, 17, 22, 23]. We aim to explore policy learning for robotic manipulation from point clouds, and the approach we propose is compatible with any architecture producing per-point outputs from point clouds. Optical flow [24, 25] and its 3D counterpart, scene flow [20, 26], are widely used in computer vision, particularly in autonomous driving setups where the objective is to associate the movement of each pixel (or a point in 3D space) from one image (or point cloud) to the next time step. We use flow as an action representation for robot manipulation, and our method could integrate prior flow estimation techniques if necessary.

Policy Learning from Point Clouds or Flow for Robotic Manipulation Learning from point clouds has been explored in grasping [19, 27], in-hand manipulation (by voxelizing) [18], visual navigation [28], and shaping 3D deformables [29]. Our work differs in that we study tasks that involve manipulating a tool in 3D space from point cloud observations, and where we use tool flow as the action representation to improve learning. While Qin et al. [30] extract tool point clouds and learn keypoints for grasping and manipulating tools, we instead predict dense tool flow for manipulating the tool. Some prior work has explored policy learning using *flow* for robot manipulation, such as for fabric folding [31], manipulating articulated objects [32, 33], and manipulating 3D deformables [34]. This work differs in that we propose a more general framework that does not assume a specific structure of the objects being manipulated, and which predicts flow on the tool a robot controls instead of flow on a target object. Furthermore, unlike prior work [35] which iteratively minimizes flow with pick and place actions, or other work [36] which uses optical flow on tactile sensors, we use flow to derive continuous tool motions in 3D space from visual input. A recent work [37] estimates optical flow using RGBD images from the current frame to the demon-

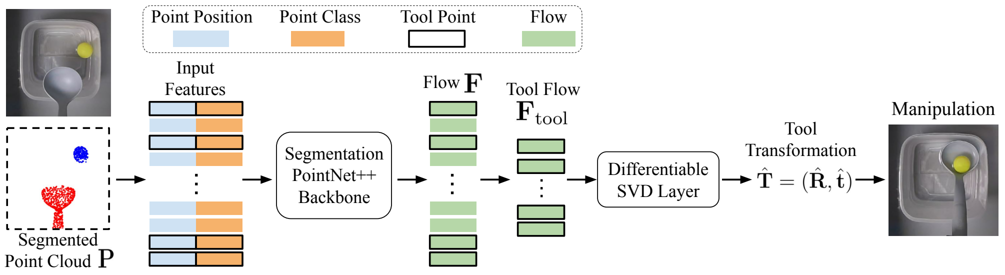
Point Position
Point Class
Tool Point
Flow
Input Features
Segmentation PointNet++ Backbone
Flow $\mathbf{F}$
Tool Flow $\mathbf{F}_{\text{tool}}$
Differentiable SVD Layer
Tool Transformation $\hat{\mathbf{T}} = (\hat{\mathbf{R}}, \hat{\mathbf{t}})$
Manipulation
Segmented Point Cloud $\mathbf{P}$

Figure 2: The proposed ToolFlowNet framework learns from segmented point clouds, which form the input to a dense point cloud network to produce per-point flow vectors. We extract just the tool points (bolded above for clarity) and use those tool points to determine the transformation that the robot should apply to the tool. See Section 3 for further details. Above, we show the physical scooping task; see Section 4.4 for details.

stration and extracts a transformation to align them. In contrast, we do not use flow for aligning frames to demonstrations but for deriving the transformations the tool should follow.

Deformable Object Manipulation We apply our proposed tool flow framework on tasks with continuous control of a tool for deformable object manipulation. Such manipulation is challenging for robots for reasons such as the difficulty in specifying a concise state representation for deformables and their complex dynamics [11, 12]. We test on scooping and pouring. Variants of these tasks have been studied in prior work. For example, [38] use scooping as an example application for task and motion planning, and [39] test scooping of granular media using a 2D image representation. Unlike these works, our approach is a general framework for robots performing continuous control of a tool to manipulate deformables in 3D space. Prior works [40, 41, 42, 43, 44] propose methods to detect or model physics properties of granular media or liquids and test on scooping and pouring. In contrast, we propose a general method of predicting 3D tool flow which does not require modeling properties of deformables and is not specialized to scooping or pouring tasks, and which uses point clouds as inputs instead of RGB images [45].

3 Method: ToolFlowNet for Behavioral Cloning from Point Clouds

We consider policy learning from segmented point cloud observations. A segmented point cloud $\mathbf{P}_t$ at time t is an $N \times d$ array with N points, each with feature dimension d. The feature of the ith point $p^{(i)} \in \mathbf{P}_t$ consists of its 3D coordinate position and a one-hot vector indicating the class of the object to which $p^{(i)}$ belongs. For ease of notation, we suppress the time subscript t and the point index superscript i when the distinction is not needed. We study Behavioral Cloning (BC) [46] from segmented point clouds. BC assumes access to a dataset $\mathcal{D} = \{(\mathbf{o}_t, \mathbf{a}_t^*)\}_{t=1}^{M}$ of observation-action pairs $(\mathbf{o}_t, \mathbf{a}_t^*)$ from a demonstrator, where $\mathbf{o}_t$ indicates any type of observation (of which segmented point clouds are one example). BC performs supervised learning to learn a policy π with parameters θ such that the predicted action $\hat{\mathbf{a}}_t = \pi_\theta(\mathbf{o}_t)$ is close to the ground truth label $\mathbf{a}_t^*$. While prone to compounding errors at test time [47], BC has shown surprising effectiveness when compared to more complex learning-based algorithms in some robotic manipulation contexts [48, 49], which motivates further study on how it can be done with point cloud observations. In future work, we will explore combining our method with other imitation learning algorithms [50, 51, 52].

3.1 Tool Flow As Action Representation

We propose to use *tool flow* as an internal representation for the action, where the flow $f^{(i)} \in \mathbb{R}^3$ associated with point $p^{(i)}$ is a 3D vector. For a given tool point, we interpret its flow vector as representing how the point will move in 3D space as a result of "applying" the flow. To form the policy π_θ, we use a dense point cloud network (such as a segmentation PointNet++ [15]), which given an input point cloud $\mathbf{P}$ produces per-point outputs. The point cloud input is already segmented in that it contains, for each point, the 3D world position and a one-hot encoding of its class. With an $(N \times d_1)$-sized point cloud $\mathbf{P}$ as input, the output $\mathbf{F}$ has dimension $(N \times d_2)$, where d_2 is the output dimension (in our case, $d_2 = 3$). We then extract from the output $\mathbf{F}$ the subset of $N' \leq N$

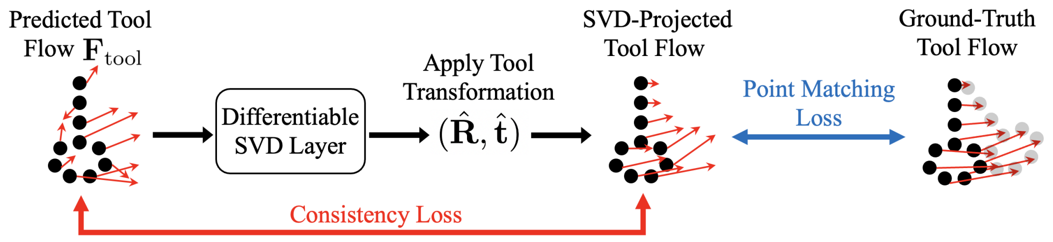
Predicted Tool Flow $\mathbf{F}_{\text{tool}}$
Differentiable SVD Layer
Apply Tool Transformation
$(\hat{\mathbf{R}}, \hat{\mathbf{t}})$
SVD-Projected Tool Flow
Point Matching Loss
Ground-Truth Tool Flow
Consistency Loss

Figure 3: A visualization of the proposed point matching loss (Eq. 3) and consistency loss (Eq. 4). The black points visualize a simplified ladle's point cloud, and the thin red arrows represent the flows on the tool points.

points in $\mathbf{P}$ corresponding to all points on the *tool*, while ignoring points belonging to other object classes. This results in a set of predicted 3D tool flow vectors $\mathbf{F}_{\text{tool}} = \{f^{(i)}\}_{i=1}^{N'}$ with $f^{(i)} \in \mathbb{R}^3$ for each tool point.

Suppose that, at time t, the expert applies an action to the tool which is given by a ground-truth transformation $\mathbf{a}^* = (\mathbf{R}^*, \mathbf{t}^*) = \mathbf{T}^* \in SE(3)$. Let $\mathbf{P}_{\text{tool}} \subseteq \mathbf{P}$ be the set of 3D points on the tool. Then the ground-truth tool flow is given by $\mathbf{F}_{\text{gt}} = \mathbf{T}^*\mathbf{P}_{\text{tool}} - \mathbf{P}_{\text{tool}}$ where $\mathbf{T}^*\mathbf{P}_{\text{tool}}$ is the result of applying the transformation $\mathbf{T}^*$ on all points in $\mathbf{P}_{\text{tool}}$. Thus, there is a one-to-one correspondence between the transformation $\mathbf{T}^*$ and flow $\mathbf{F}_{\text{gt}}$; nonetheless, we show in Section 4 that estimating the tool flow leads to improved performance compared to estimating the transformation $\mathbf{T}^*$ directly.

Given the set of predicted 3D tool flow vectors $\mathbf{F}_{\text{tool}}$, the next step is to extract a single overall action $\hat{\mathbf{a}}$, where $\hat{\mathbf{a}}$ is a transformation that represents the change in translation and rotation of the tool's pose. To compute the action, we consider the tool point clouds $\mathbf{P}_{\text{tool}}$ and $\mathbf{P}'_{\text{tool}} = \mathbf{P}_{\text{tool}} + \mathbf{F}_{\text{tool}}$, where in the latter, we move each point based on its estimated flow. Our objective is to estimate the best-fit tool transformation $\hat{\mathbf{T}} = (\hat{\mathbf{R}}, \hat{\mathbf{t}})$ which contains rotation and translation components, respectively, to align $\mathbf{P}_{\text{tool}}$ and $\mathbf{P}'_{\text{tool}}$, i.e., we want to find $\hat{\mathbf{T}}$ to minimize $||\hat{\mathbf{T}}\mathbf{P}_{\text{tool}} - \mathbf{P}'_{\text{tool}}||_2$. To obtain the rotation $\hat{\mathbf{R}}$, we first center the two tool point clouds to obtain $\bar{\mathbf{P}}_{\text{tool}}$ and $\bar{\mathbf{P}}'_{\text{tool}}$. We then input the centered point clouds to a differentiable, parameter-less Singular Value Decomposition (SVD) layer [53, 54] which computes the rotation which best aligns $\bar{\mathbf{P}}_{\text{tool}}$ and $\bar{\mathbf{P}}'_{\text{tool}}$ with respect to mean square error (MSE). The change in translation $\hat{\mathbf{t}}$ can then be computed as $\hat{\mathbf{t}} = C(\mathbf{P}'_{\text{tool}}) - \hat{\mathbf{R}}C(\mathbf{P}_{\text{tool}})$, where $C(\mathbf{P})$ denotes the centroid of the point cloud $\mathbf{P}$. By combining the translation and rotation components, we produce the full transformation $\hat{\mathbf{T}}$, which we treat as our action representation for the policy. The outputs for the non-tool points are not supervised. We call the resulting point cloud-to-action system as ToolFlowNet (see Figure 2), which can be used by a robot for manipulation. Mathematically, let F_θ represent the segmentation PointNet++ that generates the flow vectors. ToolFlowNet computes the tool transformation as follows:

$$\hat{\mathbf{a}} = (\hat{\mathbf{R}}, \hat{\mathbf{t}}) = \pi_\theta(\mathbf{o}) = \text{SVD}(\mathbf{F}_{\text{tool}}) \tag{1}$$

where SVD represents the parameter-less Singular Value Decomposition layer as described above, and $\mathbf{F}_{\text{tool}}$ is the flow corresponding to the tool points in the estimated flow $\mathbf{F} = F_\theta(\mathbf{P})$.

3.2 Imitation Learning via Tool Point Matching and Consistency Losses

Point Matching Loss: Given the policy's predicted action $\hat{\mathbf{a}} = \pi_\theta(\mathbf{o})$, a straightforward way to imitate the ground truth action $\mathbf{a}^* = (\mathbf{R}^*, \mathbf{t}^*)$ is to use the MSE loss:

$$L_{\text{mse}}(\hat{\mathbf{a}}, \mathbf{a}^*) = \beta_1||\hat{\mathbf{R}} - \mathbf{R}^*||_2 + \beta_2||\hat{\mathbf{t}} - \mathbf{t}^*||_2, \tag{2}$$

where β_1 and β_2 are weights for the translation and rotation components. Instead of trying to balance the weights, in this paper, we use a point matching loss to reduce the discrepancy between $\hat{\mathbf{a}} = (\hat{\mathbf{R}}, \hat{\mathbf{t}})$ and the ground truth action $\mathbf{a}^* = (\mathbf{R}^*, \mathbf{t}^*)$. Given the predicted action, the transformation $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ is applied on all the original N' tool points in the point cloud, and the loss function L_{point} computes the Euclidean distance between the tool points transformed using the predicted action

$(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ versus the tool points transformed using the ground-truth action $(\mathbf{R}^*, \mathbf{t}^*)$:

$$L_{\text{point}}(\mathbf{P}, \hat{\mathbf{a}}, \mathbf{a}^*) = \frac{1}{N'} \sum_{i=1}^{N'} \|(\hat{\mathbf{R}} p^{(i)} + \hat{\mathbf{t}}) - (\mathbf{R}^* p^{(i)} + \mathbf{t}^*)\|_2, \tag{3}$$

where $p^{(i)}$ iterates through the N' tool points in $\mathbf{P}_{\text{tool}} \subseteq \mathbf{P}$, and we interpret $\hat{\mathbf{R}}$ and $\mathbf{R}^*$ as representing 3×3 rotation matrices. Prior work on 6D pose estimation [55, 56, 57] has used variants of this loss function to jointly optimize for translation and rotation as compared to balancing the weights on separate translation and rotation losses. Our usage of L_{point} is similar to that in Wang et al. [19] where the matching loss is on tool points directly controllable by the robot.

Consistency Loss: While L_{point} should allow the policy π_θ to learn SE(3) pose changes (and thus, actions), its effect on optimizing the predicted flow vectors $\mathbf{F}$ happens via backpropagating through a differentiable SVD layer which "compresses" all predicted flow vectors to produce a single transformation $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$. This compression means that there could be significant noise in the individual flow vectors, even if they combine to form a reasonable action. Thus, we propose a consistency loss $L_{\text{consistency}}$ which serves as a regularizer to ensure that the *predicted* flow vectors are similar to their *induced, SVD-projected* flow vectors produced from the transformation encoded in $\hat{\mathbf{a}}$. The loss is:

$$L_{\text{consistency}}(\mathbf{P}, \hat{\mathbf{a}}) = \frac{1}{N'} \sum_{i=1}^{N'} \|(\hat{\mathbf{R}} p^{(i)} + \hat{\mathbf{t}} - p^{(i)}) - f^{(i)}\|_2, \tag{4}$$

where for each of the N' tool points, we compute $\hat{\mathbf{R}} p^{(i)} + \hat{\mathbf{t}} - p^{(i)}$ as the induced flow from the predicted transformation $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ after applying the SVD layer, and $f^{(i)}$ is the flow predicted by the network before the SVD layer. Note that the ground truth transformation $(\mathbf{R}^*, \mathbf{t}^*)$ does *not* appear in this consistency loss. The consistency loss is only a function of a set of points and a set of corresponding flow vectors on those points, and does not rely on any other ground truth signal. We combine this with the point matching loss L_{point} to obtain the final loss function to optimize the policy π_θ: $L_{\text{combo}} = L_{\text{point}} + \lambda \cdot L_{\text{consistency}}$, with hyperparameter λ controlling the weight of the consistency loss, which we set to $\lambda = 0.1$. See Figure 3 for visuals. To distinguish our method from traditional optical flow and scene flow methods, ToolFlowNet uses flow as a representation to compute the transformation of the tool, and is trained using the ground-truth demonstration action. It is not used to just estimate the flow.

Additional Implementation Details: To obtain ground truth tool flow $\mathbf{F}_{\text{gt}}$ in simulation, we determine the 3D movement of each tool point as a result of applying the demonstrator's action to transform those points. In physical settings, we scan the tool to obtain a 3D model, from which we extract tool point clouds $\mathbf{P}_{\text{tool}}$. We perform a similar calculation where we detect the transformation executed by the robot and apply it to obtain the flow for each tool point. This method of extracting $\mathbf{F}_{\text{gt}}$ only requires access to the current observed point cloud and the corresponding action. In particular, it does *not* require the perhaps more restrictive assumption of requiring one-to-one point correspondence between two consecutive point cloud observations. In addition, this method to detect flow means that it reflects the "intended" action from the robot, which may differ from the true positions of the tool points in 3D space after the robot executes the action; for example, when a collision happens with a wall, the tool points might not move, even though the robot intended for them to move. We leave alternative techniques to extract tool flow to future work.

4 Experiments

4.1 Simulation Experiments

We build on top of SoftGym [58], which provides a suite of deformable manipluation tasks and uses NVIDIA FleX [59] as the underlying physics engine. We use the simulator to obtain ground-truth segmentation labels. For the tool, we use the "observable" point cloud at each time step, so there may be occlusions. We test two tool-based simulation tasks, PourWater and ScoopBall, and for each, test two action spaces: 3D and 6D for PourWater, and 4D and 6D for ScoopBall. In PourWater, the agent

Method	Loss	Dense PN++?	N. Success ScoopBall 4D	N. Success ScoopBall 6D	N. Success PourWater 3D	N. Success PourWater 6D	Average N. Success
PCL Direct Vector	MSE	✗	0.544±0.03	0.848±0.05	0.530±0.08	0.402±0.04	0.581
PCL Direct Vector	PM	✗	0.228±0.12	0.048±0.04	0.132±0.07	0.088±0.04	0.124
PCL Dense Transformation	MSE	✓	0.519±0.07	0.824±0.06	0.539±0.05	0.344±0.03	0.556
PCL Dense Transformation	PM	✓	0.367±0.07	0.360±0.10	0.583±0.03	0.049±0.02	0.340
D Direct Vector	MSE	✗	0.190±0.07	**0.952±0.02**	0.035±0.01	0.069±0.02	0.311
D+S Direct Vector	MSE	✗	0.734±0.11	**0.928±0.03**	**0.777±0.03**	0.304±0.03	0.686
RGB Direct Vector	MSE	✗	0.354±0.05	0.776±0.05	0.698±0.02	0.324±0.05	0.538
RGB+S Direct Vector	MSE	✗	0.671±0.07	**0.944±0.02**	**0.804±0.04**	0.353±0.03	0.693
RGBD Direct Vector	MSE	✗	0.418±0.10	0.920±0.02	0.733±0.07	0.353±0.02	0.606
RGBD+S Direct Vector	MSE	✗	0.734±0.10	**0.968±0.02**	**0.830±0.03**	0.481±0.03	0.753
ToolFlowNet, No Skip Conn.	PM+C	✓	0.987±0.08	0.304±0.06	0.000±0.00	0.000±0.00	0.323
ToolFlowNet, MSE after SVD	MSE+C	✓	0.089±0.04	0.792±0.09	0.494±0.02	**0.913±0.05**	0.572
ToolFlowNet, PM before SVD	PM	✓	0.785±0.08	0.880±0.05	0.618±0.04	0.677±0.05	0.740
ToolFlowNet, No Consistency	PM	✓	0.861±0.06	0.744±0.12	0.468±0.10	0.609±0.06	0.670
ToolFlowNet (Ours)	PM+C	✓	**1.152±0.07**	**0.952±0.02**	**0.795±0.05**	0.667±0.03	**0.892**

Table 1: Behavioral Cloning (BC) results in simulation. The first 10 rows are baselines, the next 4 are ablations of our method, and the last row is our method. We report the loss functions used as MSE only, PM only (the loss in Eq. 3), or if it also uses a consistency loss (+C, from Eq. 4). We also show whether the method uses a segmentation PointNet++ (i.e., a dense architecture), and the *normalized* success rates (N. Success) across all tasks over 5 independent BC runs. The last column averages the success across the four columns. We bold the best numbers in the columns, plus any with overlapping standard errors.

controls a box which contains water and must pour the water into a fixed target box. In ScoopBall, the agent controls a ladle and needs to scoop a ball. See Appendix A.1 for more details.

4.1.1 Baseline Methods

We compare the proposed method with the following baselines (see Section 4.3 for ablations):

- **PCL Direct Vector**. Uses a *classification* PointNet++ network to directly estimate a vector action (with a translation and an axis-angle rotation). We test two variants, one which supervises with the MSE loss and another which uses the Point Matching (PM) loss from Eq. 3 on tool points.
- **PCL Dense Transformation**. Uses a *segmentation* PointNet++, and directly predicts per-point 6D vectors (translation and axis-angle). Each point cloud has a designated point as the center of rotation for the tool, and we use the output corresponding to that point as the vector action. The outputs for the other points are not supervised. This baseline is designed to isolate any benefits from using the segmentation version of PointNet++ instead of classification. As with Direct Vector, we test two variants, with supervising using the MSE or PM losses.
- **{D, RGB, RGBD} Direct Vector**. Processes images and uses a Convolutional Neural Network to directly predict an action vector (translation and axis-angle) and supervises with MSE. The inputs are either a depth image (D), the RGB image, or an RGBD image.
- **{D+S, RGB+S, RGBD+S} Direct Vector**. These are the same as the prior set of methods, except that the input images have additional channels corresponding to binary segmentation masks. We denote these new input images as: D+S, RGB+S, and RGBD+S. We include these baselines for a fairer comparison due to assuming segmentation information in the point cloud observations.

4.1.2 Experiment Protocol and Evaluation

For each task, we use a scripted demonstrator to generate a fixed set of training demos and keep the successful ones for Behavioral Cloning. We standardize on 500 training epochs for all methods and average across 5 seeds. We evaluate every 25 training epochs on 25 testing configurations and use the maximum success (averaged over 5 seeds) across the full training history, then divide this by the demonstrator success rate to get the normalized performance. See Appendix A.2.2 for more details.

4.2 Simulation Results and Analysis

The results in Table 1 suggest that using ToolFlowNet outperforms the baselines on average across the tasks. In particular, for ScoopBall 4D and PourWater 6D, it outperforms all other baselines, and

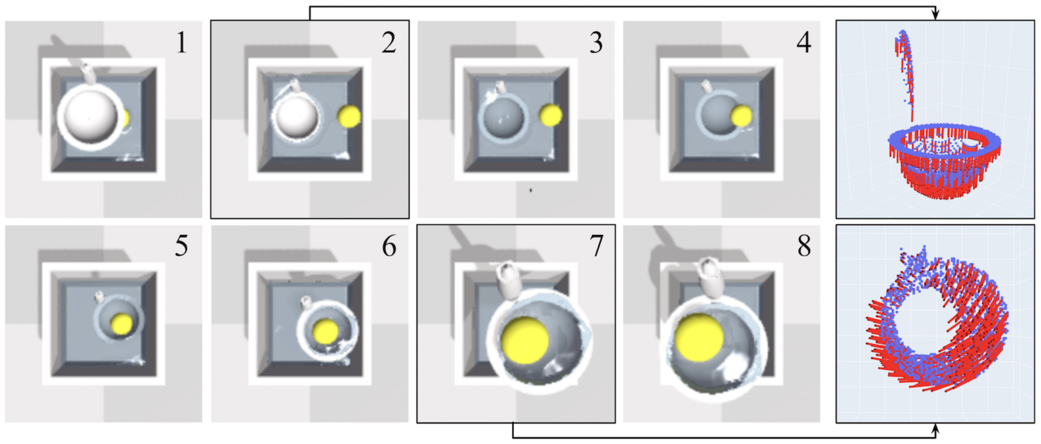

Figure 4: An example successful ScoopBall 4D rollout by a learned ToolFlowNet Behavioral Cloning policy. We show subsampled RGB frames for visual clarity, though the policy only uses point clouds as input. For two of the frames, we show the policy's tool flow visualizations. The policy lowers the ladle (frames 1-3), rotates and moves it in the direction of the ball (frames 4-5), lifts the ball (frame 6) and then rotates back to the starting pose (frames 7-8). The policy's flow visualizations for frames 2 and 7 suggest the ability to learn downward and rotation movement, respectively. The predicted flow vectors, colored red, are slightly enlarged for clarity.

for ScoopBall 6D and PourWater 3D, it is on par with the best image-based baselines. This may indicate that some tasks have a 3D nature which makes it more natural to learn policies from point clouds. Figure 4 shows a qualitative example test-time rollout of ScoopBall 4D from the learned ToolFlowNet policy. Figure 4 also visualizes the policy's internal flow predictions (i.e., the per-point flow vectors $f^{(i)}$ before the SVD layer), showing that the network has learned surprisingly clean per-point tool flow vectors. Furthermore, as the agent controls the ladle at its upper tip, when rotating, the flow vectors also correctly predict longer flow vectors for the points located further away from the origin of the tool pose.

4.3 Why Does ToolFlowNet Help Robot Learning?

We perform further experiments to determine why ToolFlowNet outperforms the baselines that directly regress to a transformation. Specifically, we create a variant of ScoopBall in which the action space consists of translations only (no rotations); see Appendix B.1 for details. These experiments reveal that ToolFlowNet does not outperform the baselines in translation-only settings, indicating that the benefits of ToolFlowNet come from predicting rotations. We also test Direct Vector methods with 4D (quaternions), 6D [60], 9D (rotation matrices) [54], and 10D [61] rotation representations in Appendix B.12, and find that ToolFlowNet continues to obtain higher success rates.

We next study ablations of ToolFlowNet to understand which components are most critical:

- **ToolFlowNet, No Skip Connections**: removes skip connections in the segmentation PointNet++.
- **ToolFlowNet, MSE after SVD**: tests applying an MSE loss on the induced transformation from SVD instead of point matching. We still use the consistency loss (Eq. 4).
- **ToolFlowNet, Point Matching (PM) Before SVD**: tests using the PM loss (Eq. 3) before the SVD layer, so the loss does not back-propagate through the SVD layer.
- **ToolFlowNet, No Consistency**: tests removing the consistency loss (and just using Eq. 3).

We use the same experiment protocol as in Section 4.1.2 on all tasks. The results, also in Table 1, suggest strong benefits to using the point matching loss, the consistency loss, and the standard segmentation PointNet++ with skip connections. For example, across all tasks, ToolFlowNet performance is worse without using a consistency loss. The utility of some design choices may be more task-specific; removing skip connections leads to no successes on PourWater because removing it made the model unable to predict any rotations (see Appendix B.4 for additional analysis), while it is possible to succeed in ScoopBall without using rotations.

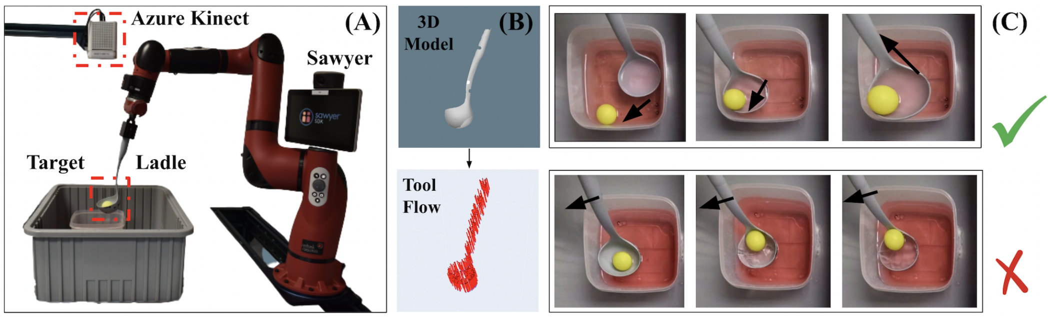

Figure 5: Physical experiments. (A) The Sawyer holds a ladle above a small box with water, which is enclosed in a larger gray box to contain spills. (B) The scanned 3D model of the ladle with a representative tool flow visualization. (C) Example test-time trials with subsampled frames. Top row: successful tool movement towards and lifting the target. Bottom row: collision failure due to repeatedly pushing against the container.

4.4 Physical Scooping Experiments

We test ToolFlowNet in the real world using a Sawyer robot with a standard consumer ladle which we scan to obtain a 3D model, and a yellow ping-pong ball acting as the target item (see Figure 5). The ladle is attached to the Sawyer's end-effector. As in simulation, we obtain tool flow by recording the change in end-effector pose and applying the transformations on the tool point cloud. A Microsoft Azure Kinect camera captures top-down depth images to compute the ball's point cloud.

A human operator manually moves the Sawyer's arm via direct touch to collect 125 demonstrations, with each comprising about 20 observation-action pairs. We use 100 demonstrations for training ToolFlowNet, and use the remaining 25 for monitoring evaluation MSE. We perform 50 physical scooping trials, where each trial begins with the human dropping the ping-pong ball over the water at an arbitrary location within the inner box shown in Figure 5. The trial is classified as successful if the Sawyer raises the ball from the water surface to above the top of the smaller box. Results in Table 2 suggest that the Sawyer achieves 41/50 successes (82%), with 9 failures. All failures were due to the ladle colliding with the small box. See Appendix C for more details. In future work we will do physical experiments with more complex demonstrations.

ToolFlowNet in Real	#Trials
Successes	41/50
Failures	9/50

Table 2: Physical scooping results.

4.5 Limitations and Failure Cases

In our experiments, we obtain the ground-truth tool flow data by applying the demonstrator's actions on a set of tool points and computing the change in the resulting tool point positions. The tool points can be observed or derived via a tool model. In either case, we require access to the demonstrator's action, and future work could relax this assumption by extracting tool flow without explicit actions, such as when a human provides a video. Possible approaches include using scene flow techniques.

A limitation of ToolFlowNet is that it may be susceptible to occlusions of the tool when a model of the tool is not available. For physical scooping, we rely on a scanned model of the tool because the Sawyer's arm would occlude parts of the tool, but tool models might not always be available. In future work, we will explore ways to address occlusions such as point cloud inpainting and tracking. Finally, we test ToolFlowNet on two simulation tasks with two action spaces for each, and scooping in real. We hope to test on more diverse tasks such as cloth or rope manipulation, and to address failures from the physical experiments.

5 Conclusion

In this work, we propose a general technique for policy learning from point clouds for tool-based manipulation tasks, which we demonstrate on scooping and pouring tasks. Our method, called ToolFlowNet, predicts per-point tool flow vectors which are converted into actions. We hope this research inspires future work on learning from point clouds.

Acknowledgments

This work was supported by LG Electronics and by NSF CAREER grant IIS-2046491. We thank Brian Okorn and Chuer Pan for assistance with the differentiable SVD layer, and Mansi Agrawal, Sashank Tirumala, and Thomas Weng for paper writing feedback.

References

[1] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning Ambidextrous Robot Grasping Policies. *Science Robotics*, 4(26), 2019.

[2] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning Dexterous In-Hand Manipulation. In *International Journal of Robotics Research (IJRR)*, 2019.

[3] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter Networks: Rearranging the Visual World for Robotic Manipulation. In *Conference on Robot Learning (CoRL)*, 2020.

[4] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. TossingBot: Learning to Throw Arbitrary Objects with Residual Physics. In *Robotics: Science and Systems (RSS)*, 2019.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end Training of Deep Visuomotor Policies. In *Journal of Machine Learning Research (JMLR)*, 2016.

[6] L. Pinto and A. Gupta. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[7] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019.

[8] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft Actor-Critic Algorithms and Applications. *arXiv preprint arXiv:1812.05905*, 2018.

[9] N. Jakobi, P.Husbands, and I. Harvey. Noise and the Reality Gap: The use of Simulation in Evolutionary Robotics. In *European Conference on Advances in Artificial Life*, 1995.

[10] O. Kroemer, S. Niekum, and G. Konidaris. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. *arXiv preprint arXiv:1907.03146*, 2019.

[11] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar. Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey. In *International Journal of Robotics Research (IJRR)*, 2018.

[12] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li, J. Pan, W. Yuan, and M. Gienger. Challenges and Outlook in Robotic Manipulation of Deformable Objects. *arXiv preprint arXiv:2105.01767*, 2021.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun. PointTransformer. In *International Conference on Computer Vision (ICCV)*, 2021.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[16] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[18] T. Chen, J. Xu, and P. Agrawal. A System for General In-Hand Object Re-Orientation. In *Conference on Robot Learning (CoRL)*, 2021.

[19] L. Wang, Y. Xiang, W. Yang, A. Mousavian, and D. Fox. Goal-Auxiliary Actor-Critic for 6D Robotic Grasping with Point Clouds. In *Conference on Robot Learning (CoRL)*, 2021.

[20] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional Scene Flow. In *International Conference on Computer Vision (ICCV)*, 1999.

[21] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-Voxel CNN for Efficient 3D Deep Learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[22] W. Wu, Z. Qi, and L. Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] X. Liu, M. Yan, and J. Bohg. MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences. In *International Conference on Computer Vision (ICCV)*, 2019.

[24] B. K. Horn and B. G. Schunck. Determining Optical Flow. Technical report, USA, 1980.

[25] P. Fischer, A. Dosovitskiy, E. Ilg, P. Hausser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[26] Z. Teed and J. Deng. RAFT-3D: Scene Flow using Rigid-Motion Embeddings. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] Y.-H. Wu, J. Wang, and X. Wang. Learning Generalizable Dexterous Manipulation from Human Grasp Affordance. In *Conference on Robot Learning (CoRL)*, 2022.

[28] K. Lobos-Tsunekawa and T. Harada. Point Cloud Based Reinforcement Learning for Sim-to-Real and Partial Observability in Visual Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[29] B. Thach, B. Y. Cho, A. Kuntz, and T. Hermans. Learning Visual Shape Control of Novel 3D Deformable Objects from Partial-View Point Clouds. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[30] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. KETO: Learning Keypoint Representations for Tool Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[31] T. Weng, S. Bajracharya, Y. Wang, K. Agrawal, and D. Held. FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy. In *Conference on Robot Learning (CoRL)*, 2021.

[32] S. Pillai, M. R. Walter, and S. Teller. Learning Articulated Motions From Visual Demonstration. In *Robotics: Science and Systems (RSS)*, 2014.

[33] B. Eisner, H. Zhang, and D. Held. FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects. In *Robotics: Science and Systems (RSS)*, 2022.

[34] B. Shen, Z. Jiang, C. Choy, L. J. Guibas, S. Savarese, A. Anandkumar, and Y. Zhu. ACID: Action-Conditional Implicit Visual Dynamics for Deformable Object Manipulation. In *Robotics: Science and Systems (RSS)*, 2022.

[35] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox. IFOR: Iterative Flow Minimization for Robotic Object Rearrangement. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[36] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez. Tactile-RL for Insertion: Generalization to Objects of Unknown Geometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[37] M. Argus, L. Hermann, J. Long, and T. Brox. FlowControl: Optical Flow Based Visual Servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[38] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning Compositional Models of Robot Skills for Task and Motion Planning. In *International Journal of Robotics Research (IJRR)*, 2019.

[39] C. Schenck, J. Tompson, D. Fox, and S. Levine. Learning Robotic Manipulation of Granular Media. In *Conference on Robot Learning (CoRL)*, 2017.

[40] C. Schenck and D. Fox. Towards Learning to Perceive and Reason About Liquids. In *International Symposium on Experimental Robotics (ISER)*, 2016.

[41] C. Schenck and D. Fox. Visual Closed-Loop Control for Pouring Liquids. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[42] C. Schenck and D. Fox. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. In *Conference on Robot Learning (CoRL)*, 2018.

[43] S. Clarke, T. Rhodes, C. G. Atkeson, and O. Kroemer. Learning Audio Feedback for Estimating Amount and Flow of Granular Material. In *Conference on Robot Learning (CoRL)*, 2018.

[44] C. Matl, Y. Narang, R. Bajcsy, F. Ramos, and D. Fox. Inferring the Material Properties of Granular Media for Robotic Tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[45] G. Salhotra, I.-C. A. Liu, M. Dominguez-Kuhne, and G. S. Sukhatme. Learning Deformable Object Manipulation from Expert Demonstrations. In *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[46] D. A. Pomerleau. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Comput.*, 3, 1991.

[47] S. Ross, G. J. Gordon, and J. A. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[48] S. Dasari, J. Wang, J. Hong, S. Bahl, Y. Lin, A. Wang, A. Thankaraj, K. Chahal, B. Calli, S. Gupta, D. Held, L. Pinto, D. Pathak, V. Kumar, and A. Gupta. RB2: Robotic Manipulation Benchmarking with a Twist. *NeurIPS 2021 Datasets and Benchmarks Track*, 2021.

[49] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit Behavioral Cloning. In *Conference on Robot Learning (CoRL)*, 2021.

[50] E. Johns. Coarse-to-Fine Imitation Learning: Robot Manipulation from a Single Demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[51] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An Algorithmic Perspective on Imitation Learning. *Foundations and Trends in Robotics*, 7, 2018.

[52] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A Survey of Robot Learning From Demonstration. *Robotics and Autonomous Systems*, 57, 2009.

[53] O. Sorkine-Hornung and M. Rabinovich. Least-Squares Rigid Motion Using SVD. Technical report, ETH Zurich, 2016.

[54] J. Levinson, C. Esteves, K. Chen, N. Snavely, A. Kanazawa, A. Rostamizadeh, and A. Makadia. An Analysis of SVD for Deep Rotation Estimation. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[55] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[56] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. In *European Conference on Computer Vision (ECCV)*, 2018.

[57] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[58] X. Lin, Y. Wang, J. Olkin, and D. Held. SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. In *Conference on Robot Learning (CoRL)*, 2020.

[59] M. Macklin, M. Muller, N. Chentanez, and T.-Y. Kim. Unified Particle Physics for Real-Time Applications. *ACM Trans. Graph.*, 33(4), July 2014.

[60] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the Continuity of Rotation Representations in Neural Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[61] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly. A Smooth Representation of Belief over SO(3) for Deep Rotation Learning with Uncertainty. In *Robotics: Science and Systems (RSS)*, 2020.

[62] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[63] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[64] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep Reinforcement Learning that Matters. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[65] A. Srinivas, M. Laskin, and P. Abbeel. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2020.

[66] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[67] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501*, 2020.

[68] J. Chen, Y. Yin, T. Birdal, B. Chen, L. Guibas, and H. Wang. Projective Manifold Gradient Layer for Deep Rotation Regression. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[69] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys. Deep Q-learning from Demonstrations. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[70] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[71] L. Rozo and V. Dave. Orientation Probabilistic Movement Primitives on Riemannian Manifolds. In *Conference on Robot Learning (CoRL)*, 2021.

[72] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online Movement Adaptation Based on Previous Sensor Experiences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[73] J. Urain, D. Tateo, and J. Peters. Learning Stable Vector Fields on Lie Groups. In *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[74] Y. Huang, F. J. Abu-Dakka, J. Silverio, and D. G. Caldwell. Toward Orientation Learning and Adaptation in Cartesian Space. In *IEEE Transactions on Robotics*, 2020.

[75] M. Liu, X. Li, Z. Ling, Y. Li, and H. Su. Frame Mining: a Free Lunch for Learning Robotic Manipulation from 3D Point Clouds. In *Conference on Robot Learning (CoRL)*, 2022.

[76] Zeromq. URL `https://zeromq.org/`.