VIRDO++: Real-World, Visuo-Tactile Dynamics and Perception of Deformable Objects

Youngsun Wi[1] **Andy Zeng**[2] **Pete Florence**[2] **Nima Fazeli**[1]
[1]Robotics Department, University of Michigan [2]Robotics at Google
{yswi, nfz}@umich.edu {andyzeng, peteflorence}@google.coms
https://www.mmintlab.com/virdopp

Abstract: Deformable objects manipulation can benefit from representations that seamlessly integrate vision and touch while handling occlusions. In this work, we present a novel approach for, and real-world demonstration of, multimodal visuo-tactile state-estimation and dynamics prediction for deformable objects. Our approach, VIRDO++, builds on recent progress in multimodal neural implicit representations for deformable object state-estimation [1] via a new formulation for deformation dynamics and a complementary state-estimation algorithm that (i) maintains a belief distribution of deformation within a trajectory, and (ii) enables practical real-world application by removing the need for contact patches. In the context of two real-world robotic tasks, we show: (i) high-fidelity cross-modal state-estimation and prediction of deformable objects from partial visuo-tactile feedback, and (ii) generalization to unseen objects and contact formations.

Keywords: Deformable Object Manipulation, Multimodal Representation Learning

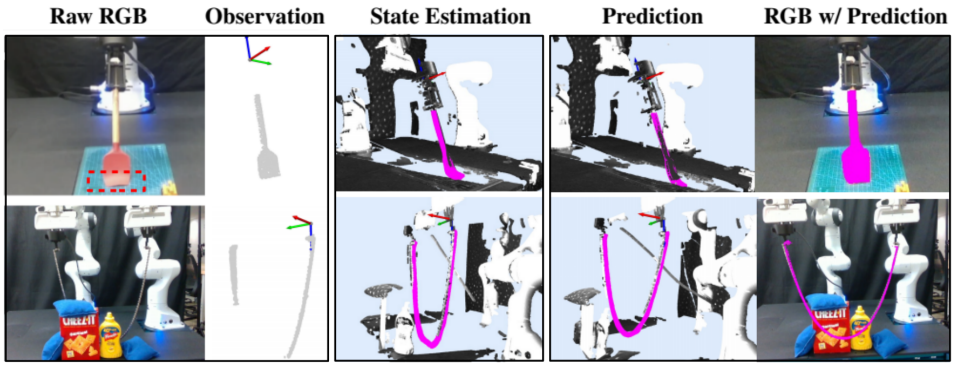

Fig. 1: Given partial point cloud observations (left), and multimodal sensor measurements (end effector wrist reaction wrench and pose), VIRDO++ can accurately predict the 3D geometry (magenta) and reaction forces of deformable objects such as spatulas (top) and bike chains (bottom) conditioned on the next robot action under severe occlusions – which can be artificial (red dotted box) or natural due to obstacles (e.g., from other objects).

1 Introduction

Deformable objects are ubiquitous in our everyday lives – the clothes we wear, the food we eat, and many of the tools we use are just a few examples. As such, helpful robots of the future may benefit from the ability to master dexterous manipulation of deformable objects. At the heart of this mastery is the interplay between object geometry and force transmission, perceived by vision and touch. Deformable object manipulation is difficult due to the complexity of this interplay (e.g., infinite dimensional state-spaces and nonlinearity) and the ensuing challenges in perception (e.g., partial observability and occlusion) and controls [2, 3, 4, 5]. Multimodal visuo-tactile representations can help address many of these challenges by exploiting mutual information and complementary cues.

Existing deformable object manipulation approaches typically use one modality (mostly vision) and rely on finite element/particle-based techniques [6, 7, 8, 9, 10, 11, 12, 13] or leverage deep learning

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

for visual affordance/latent dynamics learning [14, 15, 16, 17, 18, 19]. The former methods typically rely on privileged knowledge (e.g., occluded or unknown boundary conditions) and stop at system identification, limiting their ability to refine the underlying physics model by learning from data. Latter methods lack (i) tactile feedback which is far more informative of contact than vision, and (ii) structured representations which can limit generalization and increase sample-complexity.

Here, we build on recent work [1] that provides a framework for visuo-tactile state estimation using multimodal implicit neural representations. This approach offers a number of advantages including multimodal sensory fusion, direct integration with raw sensory feedback, and computational speed. However, [1] does not address (i) deformation predictions (dynamics) and (ii) is not directly applicable to real-world robotics settings. The first limitation is due to the lack of a dynamics module and an emphasis on static scenes. The second limitation is due to the need for the contact patch which are typically impractical to measure in real-world robotic settings.

The primary contribution of this work is (i) a novel approach to, and (ii) real-world demonstration of, performing multi-modal visuo-tactile state estimation and dynamics prediction for deformable objects. Fig. 1 shows how our proposed approach uses partial views and tactile feedback for deformation prediction and state-estimation in real-world applications. Our contributions are:

1. Representation of deformable objects *dynamics* conditioned on actions: Multimodal neural implicit representation with action-conditioned dynamics to predict future deformations. To demonstrate the utility of the dynamics model, we introduce a state estimation algorithm based on particle filtering to maintain a belief distribution over object deformations.

2. Real-world demonstrations on two challenging tasks: Scraping with a spatula and chain manipulation. We demonstrate both 3D geometry estimation in heavy occlusion and action-conditioned deformation/reaction wrench prediction. This is enabled by introducing a novel contact latent vector that eliminates the need for explicit contact patch information used in [1]. We use these tasks to demonstrate **generalization to unseen objects and novel environments**.

2 Related Work

Deformable object modeling. Recent studies have attempted to address challenges in system identification and high computation costs in conventional continuum mechanics [6, 7, 8, 9, 10, 11, 12, 13]. To ease the burden of system identification, prior studies infer the physics models' parameters based on high-fidelity physics engines [20, 21, 22] and simple force-deformation relationship (e.g. Hooke's law) [23]. The inherent limitations of these approaches are that (i) performance is confined to the underlying physics model, often with strong assumptions on the objects (uniform density/elasticity) and the force-deformation relationship (linearity), and (ii) access to privileged information such as occluded or unknown boundary conditions. More recent approaches propose computationally efficient elastic object modelings with minimum system identification using, for example, a potential energy propagation [24] and geometric motion estimates [25]. However, these methods suffer from balancing between model approximation and computation cost as well as maintaining seamless integration with robotic sensing modalities (RGB-D cameras and F/T sensors). In this study, we adopt a data-driven approach using neural networks [1, 26, 27]. Learning directly from observation, our approach does not require object parameters (Young's modulus, Poisson's ratio) or a strong assumption of object composition. Even so, it can capture complex non-linear elastic behaviors and exhibit fast computation times due to the architecture of the neural networks.

3D geometric representation of deformable objects. Most recent studies have adopted discrete geometry representation for deformable objects with finite resolution such as meshes [7, 10, 28] and key points [15, 21], sometimes accompanied with graph neural networks [29, 27]. These approaches have been used largely for 1D and 2D objects (rope, cloth, top view of cylindrical objects). Among them, structured representations (e.g., Mesh or GNN) are advantageous at tracking connectivity, mainly used for deformable objects with large self-occlusions (e.g. folded cloth or knot) [30, 31]. On the other hand, we address general 3D geometry representations which are not confined to specific types of objects and can work with large occlusions (e.g., wok-scraping). Our work builds on dense implicit geometric representation which include signed distances fields [1, 26, 32], occupancy [33],

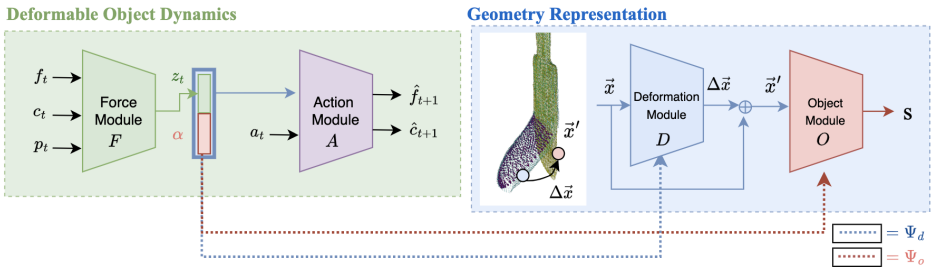

Fig. 2: **Overview.** VIRDO++ is composed of an implicit signed-distance field representation of geometry (left), informed by a deformation dynamics model (right), where dotted lines indicate hyper networks that decode embeddings into network weights.

volume density [34], and dense descriptors [35]. These approaches have several advantages over their discrete counterparts in dealing with large occlusion, high resolution surface representation [36], parameterization of 3D geometries [37, 38], and are useful for downstream tasks (e.g., contact location detection similar to Sec. 4.5). We extend these representations to real-world multimodal deformable object dynamics and perception by integrating touch and robot actions.

3 Methodology

Our approach, shown schematically in Fig. 2, is composed of a latent deformation dynamics model and an implicit dense geometric representation. Using Hidden Markov Models as an analogy, the former component plays the role of the hidden state transition, and the latter is the observation model. We expand on the latent deformation dynamics in Sec. 3.1 and discuss the implicit geometric representation in Sec. 3.2.

3.1 Deformable Object Dynamics

VIRDO++ represents deformable object states using latent object ($\boldsymbol{\alpha}$) and force ($\boldsymbol{z_t}$) codes. The object code condenses the undeformed shape information into a feature representation that can interpolate to unseen object variants. The force code is generated by the Force Module (**F**) which encodes boundary conditions as $\boldsymbol{z}_t = \mathbf{F}(\boldsymbol{f}_t, \boldsymbol{c}_t, \boldsymbol{p}_t)$ where $\boldsymbol{f}_t \in \mathbb{R}^6$ is the reaction wrench at the wrist, $\boldsymbol{p}_t \in \mathbb{R}^6$ is robot end-effector pose, and $\boldsymbol{c}_t \in \mathbb{R}^{lc}$ is the contact latent vector, where lc is the dimension of the contact latent vector. The contact latent vector $\boldsymbol{c}_t$ is a learnable latent vector that can change over time whose primary role is to store contact information. However, it can also implicitly integrate information about the object's physical properties (e.g., stiffness) and historical information about the object's states and deformations. This embedding generalizes the original VIRDO [1] in terms of both (i) no longer needing known contact patch locations, unlocking the capability to run VIRDO in the real-world, and (ii) representing additional useful information beyond just the contact location as was the case in VIRDO. In more detail, VIRDO [1] uses ground truth contact patches encoded by a PointNet encoder [39] directly which required access to this privileged information. Here, the contact patch and encoder are replaced by the learnable contact latent vector.

The Action Module (**A**) predicts the next step's boundary conditions given the current (latent) state and robot action: $\mathbf{A}(\boldsymbol{\alpha}, \boldsymbol{z_t}, \boldsymbol{a_t}) = \hat{\boldsymbol{f}}_{t+1}, \hat{\boldsymbol{c}}_{t+1}$ where $\hat{\boldsymbol{f}}_{t+1} \in \mathbb{R}^6$ is the predicted reaction wrench at the wrist, $\hat{\boldsymbol{c}}_{t+1}$ is the predicted contact latent vector, and $\boldsymbol{a}_t \in \mathbb{R}^6$ is the robot action (Cartesian displacement). This predictive capability enables state-estimation and downstream tasks such as contact location detection crucial for deformable object manipulation. Though we did not include in this paper, the predictive ability of VIRDO++ also enables planning for deformable object manipulation. VIRDO lacked this functionality and could only infer the object geometry statically.

3.2 Geometry Representation

VIRDO++ decouples the 3D geometric representation of deformable objects into an undeformed (nominal) shape signed-distance field (SDF) and a set of deformation fields, similar to [1, 38]. The deformation field is a 3D vector field that, when summed with the deformed object SDF, results in the nominal shape SDF. More precisely, any query point $\boldsymbol{x} \in \mathbb{R}^3$ in the robot wrist frame belonging to the deformed shape SDF can be mapped back to the corresponding point $\boldsymbol{x}'$ of the nominal shape SDF by applying the point-wise deformation field $\Delta\boldsymbol{x}$ as $\boldsymbol{x} + \Delta\boldsymbol{x} = \boldsymbol{x}'$.

Nominal Shape Representation: The Object Module $\mathbf{O}(\boldsymbol{x}')$ is the parametric representation of the nominal shape SDF – the object shape with no external contacts corresponding to $\Delta\boldsymbol{x} = 0$. Here, we use a feedforward neural network to parameterize the SDF [37] and write $\mathbf{O}(\boldsymbol{x}'|\boldsymbol{\Psi}_o(\boldsymbol{\alpha})) = s$ where

s is the signed-distance at query point $\boldsymbol{x}'$ and $\boldsymbol{\Psi}_o(\boldsymbol{\alpha})$ is a hyper-network that predicts the weights of $\mathbf{O}$ conditioned on the object code $\boldsymbol{\alpha}$. The object code and hyper-network weights are learned end-to-end in an auto-decoder approach [37]. We simplify our notation to $\mathbf{O}(\boldsymbol{x}')$ for the remainder.

Deformation Field Representation: The generalized geometric representation of a deformable object is given by $\text{SDF}(\boldsymbol{x}) = \mathbf{O}_{\boldsymbol{\Phi}_o}\big(\boldsymbol{x} + \mathbf{D}(\boldsymbol{x}|\Psi_d(\boldsymbol{z}_t, \boldsymbol{\alpha}_t))\big) = s$. Here, the Deformation Module $\mathbf{D}(\boldsymbol{x}|\Psi_d(\boldsymbol{z}_t, \boldsymbol{\alpha}_t)) = \Delta\boldsymbol{x}$ produces an object's deformation field given force and object code pairs. Similar to the Object Module, Deformation Module's weights are given by the hyper-network Ψ_d.

3.3 Training and Loss Formulation

The training dataset for VIRDO++ is a set of trajectories $\mathcal{S} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$, where each trajectory $\mathcal{T}_j = \{(\boldsymbol{p}_0, \boldsymbol{f}_0, \boldsymbol{P}_0, \boldsymbol{a}_0), (\boldsymbol{p}_1, \boldsymbol{f}_1, \boldsymbol{P}_1, \boldsymbol{a}_1), ..., (\boldsymbol{p}_k, \boldsymbol{f}_k, \boldsymbol{P}_k, \boldsymbol{a}_k)\}$ has k sequential observation and action tuples. Here $\boldsymbol{P}_j$ is a partial point cloud. Our first step is to pretrain a nominal shape representation (i.e., hyper-network $\boldsymbol{\Psi}_\mathbf{o}$ and object code $\boldsymbol{\alpha}$). The loss for training m nominal shapes is $\mathcal{L}_{nominal} = \mathcal{L}_{sdf} + \lambda_1\mathcal{L}_{latent} + \lambda_2\mathcal{L}_{hyper}$ as in [1], where $\mathcal{L}_{latent} = \sum_{i=1}^{m} \|\boldsymbol{\alpha}^i\|_2$, $\mathcal{L}_{hyper} = \sum_{i=1}^{m} \|\boldsymbol{\Psi}_o(\boldsymbol{\alpha}^i)\|_2$, and:

$$\mathcal{L}_{sdf} = \sum_{i=1}^{m} \Big(\sum_{\bar{\boldsymbol{x}} \in \boldsymbol{\Omega}} |clamp(\mathbf{O}^i(\boldsymbol{x}), \delta) - clamp(s^*, \delta)| + \lambda \sum_{\bar{\boldsymbol{x}} \in \boldsymbol{\Omega}_0} (1 - \langle \nabla \mathbf{O}^i(\boldsymbol{x}), \boldsymbol{n}^* \rangle) \Big).$$

$\boldsymbol{n}^*$ is the ground truth normal vector, δ is a parameter that clips SDF predictions, and s^* is the ground truth signed distance.

After the pretraining, we train the rest of the modules using the three losses $\mathcal{L}^{geo}, \mathcal{L}^{pred}$ and $\mathcal{L}^{reg}$ summed over a fixed horizon (w) while freezing Ψ_o and the object code α:

$$\mathcal{L}_{tot} = \sum_{t=t_0}^{t_0+w-1} (\mathcal{L}_t^{geo} + \lambda_3\mathcal{L}_t^{pred} + \lambda_4\mathcal{L}_t^{reg}). \tag{1}$$

The geometry representation loss is defined as:

$$\begin{aligned}\mathcal{L}^{geo} = \lambda_5 \underbrace{\sum_{\boldsymbol{x} \in \boldsymbol{\Omega}} \|\mathbf{D}_{\boldsymbol{\Psi}_d}(\boldsymbol{x})\|_2}_{\text{minimum correction}} + \lambda_6 \underbrace{\text{CD}(\boldsymbol{P} + \mathbf{D}_{\boldsymbol{\Psi}_d}(\boldsymbol{P}), \bar{\boldsymbol{P}}^*)}_{\text{correspondence}} + \lambda_7 \underbrace{\sum_{\boldsymbol{x} \in \boldsymbol{\Omega}_0} (1 - \langle \nabla_{\boldsymbol{x}} \mathbf{O}_{\Psi_o}(\boldsymbol{x}'), \boldsymbol{n}^* \rangle}_{\text{normal aligning}} \\ + \lambda_8 \underbrace{\sum_{\boldsymbol{x} \in \boldsymbol{\Omega}} |clip(\mathbf{O}_{\boldsymbol{\Psi}_o}\big(\boldsymbol{x} + \mathbf{D}_{\boldsymbol{\Psi}_d}(\boldsymbol{x}), \delta\big) - clip(s^*, \delta)|}_{\text{signed distance regression}}.\end{aligned} \tag{2}$$

where $\boldsymbol{x}' = \boldsymbol{x} + \mathbf{D}_{\Psi_d}(\boldsymbol{x})$, Ω is the 3D querying space, $\Omega_0 \subset \Omega$ is the on-surface region, $\mathbf{P} := \{\boldsymbol{p} | \boldsymbol{p} \in \Omega_0\}$ is an unordered set of the on-surface points, and $\bar{\boldsymbol{P}}^*$ is the ground truth nominal shape point cloud. Next, we add a reaction wrench prediction loss as $\mathcal{L}_{pred} = \|\boldsymbol{f}_{t+1} - \hat{\boldsymbol{f}}_{t+1}\|$. The prediction loss for $\hat{\boldsymbol{c}}_{t+1}$ is implicitly handled by Eq. 2 because the contact latent vector prediction is recursively used as next step's input during training. In contrast, we do replace $\hat{\boldsymbol{f}}_{t+1}$ with real measurement in Eq. 2. Finally, regularization losses $\mathcal{L}_t^{reg} = \lambda_9\|\boldsymbol{z}_t\|_2 + \lambda_{10}\|\boldsymbol{c}_t\|_2 + \lambda_{11}\|\boldsymbol{\Psi}_d(\boldsymbol{\alpha}, \boldsymbol{z}_t)\|_2$, which regularize the force codes, contact latent vector, and the weights of $\mathbf{D}$ respectively.

3.4 Inference and State-estimation Algorithm

To maintain a belief distribution over the object deformation, we develop a particle filter-based inference algorithm, Alg. 1. Each particle represents a contact latent vector which can be used to reconstruct the full 3D object geometry. The `Refine` step updates the particle set by performing gradient descent on $\min_{\hat{\mathbf{C}}_\mathbf{t}} \mathcal{L}^{geo}(\boldsymbol{P}_t, \hat{\mathbf{C}}_t, \boldsymbol{f}_t, \boldsymbol{p}_t)$ with the visible object point cloud. The particles are then weighted according to their wrench error prediction with `Weight-Function` defined as $\boldsymbol{w}_t^i = exp(-\gamma(\hat{\boldsymbol{f}}_t^i - \boldsymbol{f}_t))$ and resampled with probability proportional to this weight vector. Here, $\hat{\boldsymbol{f}}_\boldsymbol{t}^\boldsymbol{i}$ is the reaction wrench prediction of the ith particle and $\boldsymbol{f}_\boldsymbol{t}$ is a ground truth measurement. Finally, the particles are propagated through the Force and Action Modules, a small amount of uncertainties is applied $\sim \mathcal{N}(0, 0.01)$ to the particles and the algorithm is repeated. (see algorithm variations in **Appendix** Sec. A.2.2)

Algorithm 1 Particle Filter based Inference Algorithm

procedure
  $\hat{\mathbf{C}}_0 \leftarrow \{\hat{\boldsymbol{c}}_0^0, \hat{\boldsymbol{c}}_0^1, ..., \hat{\boldsymbol{c}}_0^{n-1}\} \sim \mathcal{N}(0, 0.01)$ ▷ Init Particles
  for each $scan$ in $Trajectory\ Length$ **do**
    $\mathbf{C}_t \leftarrow \texttt{Refine}(\mathbf{P}_t, \hat{\mathbf{C}}_t, \boldsymbol{f}_t, \boldsymbol{p}_t)$
    $\boldsymbol{w}_t \leftarrow \texttt{Weight-Function}(\boldsymbol{f}_t, \hat{\boldsymbol{f}}_t^0, \hat{\boldsymbol{f}}_t^1, ..., \hat{\boldsymbol{f}}_t^{n-1})$
    $\bar{\mathbf{C}}_\mathbf{t} \leftarrow \texttt{Re-sampling}(\boldsymbol{w}_t, \mathbf{C}_t)$
    $\mathbf{Z}_\mathbf{t} \leftarrow \texttt{Force-Module}(\bar{\mathbf{C}}_t, \mathbf{f}_t, \mathbf{p}_t, \alpha)$
    $\{(\hat{\boldsymbol{f}}_{t+1}^0, \hat{\boldsymbol{c}}_{t+1}^0), (\hat{\boldsymbol{f}}_{t+1}^1, \hat{\boldsymbol{c}}_{t+1}^1), ..., (\hat{\mathbf{f}}_{t+1}^{n-1}, \hat{\boldsymbol{c}}_{t+1}^{n-1})\} \leftarrow$ Action-Module $(\mathbf{Z}_\mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{a}_t)$
    $t \leftarrow t + 1$
  end for
end procedure

4 Experiments

For real robot experiments, we investigate two classes of objects: **spatulas** and **bike chains**. These objects exhibit different deformation behaviors: (i) spatulas are representative of bendable elastic objects that retain their shape when external forces are removed, whereas (ii) bike chains (with the 2 end points controlled by a bi-manual manipulator) can undergo significantly larger deformations (e.g., from gravity) – requiring algorithms that can reason over free-space while estimating in-contact shapes and forces. Spatula deformations are generally local and experience significant reaction wrenches, while chain deformations are much larger but with much smaller reaction wrenches.

4.1 Spatula Manipulation Dataset

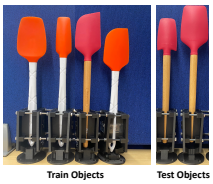

Fig. 3: Train and test objects differ in terms of (i) geometry, and (ii) how they react under wrench.

Train and test objects are shown in Fig. 3. The training set consists of 4 spatulas, each with 672 samples ($N = 28$, $k = 24$), using 9:1 train/test split (25 train and 3 test trajectories) for representation learning. The test dataset consists of 2 held-out spatulas, each with 72 samples ($N = 3$, $k = 24$), and is used to evaluate zero-shot transfer and not used for representation learning. At the beginning of each trajectory ($t = 0$), the end-effector moves to a random world Cartesian position in a bounding box of size $(0.1 \times 0.24 \times 0.06)$ meters and the tool is brought into contact with the table. Within a trajectory, the tool maintains contact while sampling a random action from $(x, y, z, r, p, y) = \pm(0.02, 0.02, 0.01, 0.05, 0.04)$ where translations are in meters and rotations in radians. Wrench measurements are averaged over 0.05 [s] and the point cloud is registered to the wrist frame. The method only requires a full point cloud from the nominal geometry of the object. This is collected by holding the object in front of the camera and rotating. Deformed object geometries are collected from a single camera viewing the front side of the spatulas without occlusions and are all partial. $\boldsymbol{P}$ is then the result of segmenting the object pointcloud and normalizing the scale into a bounding box of size $2 \times 2 \times 2[m]$ centered around $(0, 0, 0)$. The test dataset point clouds $\boldsymbol{P}$ contain occlusions at the bottom $0.15 \times$ `spatula-height` as shown in the top observation panel in Fig. 1. For hardware, we use Franka Emika's Panda, wrist mounted ATI Gamma F/T sensor, and Photoneo PhoXi 3D L.

4.2 State Estimation and Predictions

Here, we evaluate the deformation prediction (Sec. 3.1) and state-estimation (Alg. 1) accuracy of VIRDO++ on training and test trajectories. Fig. 4 shows an example of state-estimation for a test trajectory where the ground truth (black) and reconstruction (magenta) are overlaid. We note the high state-estimation accuracy indicated by the overlay agreement, where the average distance between the measured points and reconstructions is within

CD ($\times 10^3$)	Spatula w/ table	Bike Chain
Train Est.	0.777 (0.215)	0.708 (0.331)
Train Pred.	0.830 (0.330)	0.762 (0.224)
Test Est.	0.934 (0.210)	0.760 (0.234)
Test Pred.	1.041 (0.348)	1.247 (0.391)

Table. 1: VIRDO++ can accurately estimate object geometry deformations under significant occlusions, measured in terms of mean Chamfer distances (CD) (std.) $[m^2]$ scaled as $(\times 10^3)$ across 5600 on-surface points between predicted and ground truth reconstructions.

$0.03[m]$ of normalized scale, just 1% of the object length. The quantitative results, Tab. 1, show VIRDO++ is able to faithfully predict and estimate object deformations and reaction forces despite significant occlusions. Force and torque norm errors are less than 0.567 [N], 0.106 [Nm] (train) and 1.130 [N], 0.135 [Nm] (test) respectively (see **Appendix** Sec. A.3.3 and Sec. A.3.4 for details).

Our ablation study, Tab. 3, helps further anchor the scale of these state-estimation and dynamics prediction errors. On state estimation, for example, our full VIRDO++ method achieves a Chamfer distance ($\times 10^3$) of normalized geometry as 0.93 $[m^2]$. The baseline model without a novel contact latent vector input has 27% increased chamfer distance error as 1.18 $[m^2]$ for geometry estimation. Without c_t, the baseline does not reason about contact during inference, resulting in larger errors during state-estimation. Further if we assume the object is rigid and reduce our state estimation to a "rigid pose estimation" pipeline, the Chamfer distance ($\times 10^3$) increases considerably to 16.19 $[m^2]$.

The geometry predictions are slightly less accurate than state-estimation because the geometry predictions are computed by the learned dynamics using previous observation and action, while state-estimations use the current measurement to update the prediction. Despite this, the prediction Chamfer distances are within 14% of the state estimation results. Moreover, the train and test trajectories results show 25% difference in Chamfer distance. This is interesting because the test trajectories are not only from unseen tool configurations but also with occlusions (see **Dataset**). This suggests that learned dynamics of VIRDO++ generalize effectively to unseen contact formations with occlusions.

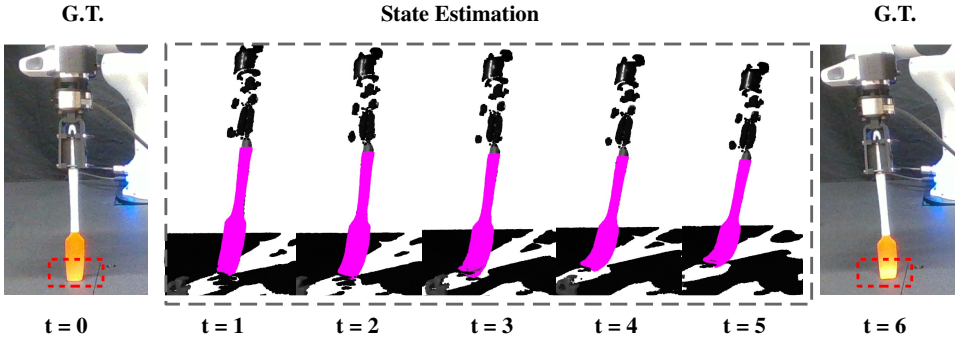

Fig. 4: Even under occlusions (generated by cropping out data within the red dotted box), our approach can accurately infer the how the geometry of a test spatula deforms over time (t = 0 to 6), indicated by high overlap agreement between the prediction (magenta) overlaid on the ground truth (black).

4.3 Evaluation on a Downstream Manipulation Task - Extrinsic Contact Detection

Central to compliant tool manipulation is estimating and controlling the contact between the tool and environment. For example, when scraping a wok with a spatula, the robot must reason over the contact formation (e.g., point, line, or patch) to ensure the spatula is scraping properly at the desired region on the wok. In this section, we evaluate VIRDO++ ability to estimate extrinsic contact features; specifically, the contact line created as the spatula is brought into contact with the table. To this end, we compute the signed distance values of the tables surface $(z = 0)$ to compute a contact cluster. This requires only a single feed-forward path where the input is an $N \times 3$ matrix, where N is the number of query points. From the contact cluster, we extract a contact line by selecting the two points furthest apart similar to [40, 41, 42]. The evaluation metrics are the average Euclidean distance $\frac{1}{2}\Sigma_{i=0}^{1}||l_i - l_i^*||$, where l_i is the detected contact lines, l_i^* is the ground truth contact lines, and $i = 0, 1$ refers to right and left ends of each contact line respectively. Ground truth contact lines were obtained from ground truth pointcloud without occlusions by applying a contact mask $z \leq \epsilon$. This contact feature estimation task is particularly interesting as it is the inverse problem of [1]; estimating [1]'s privileged input (=contact location) by inferring c_t.

As shown in Tab. 2, VIRDO++ can accurately detect contact locations with less than 8 mm error on a flat table for the current and the following time step regardless of occlusions at the tip. As in Sec.4.2, test trajectories from 4 spatulas with 15% occlusion were used for the evaluation. Tab.2 shows that the average contact line detection errors from all 4 spatulas are less than 7 [mm] during estimation and are less than 8 [mm] for predictions in Euclidean error. Similar to the result in

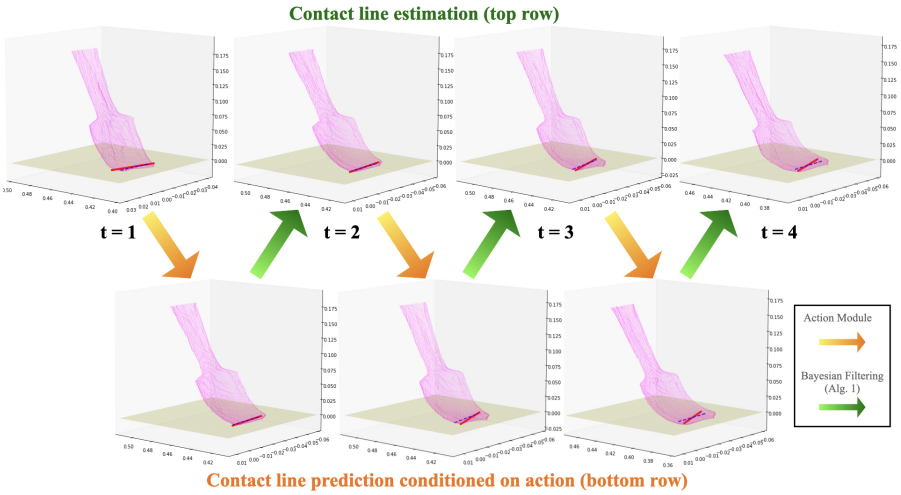

Fig. 5: VIRDO++ detects the line of contact throughout an alternating sequence of "estimation → prediction → estimation → ...". Bayesian filtering increase the accuracy of contact location detection as the red lines (detected contact line) gets closer to the dotted blue lines (ground truth) following the green arrow. In each figure, yellow plane: table location, magenta spatula: reconstructed geometry from VIRDO++ , red line: detected contact line, blue: ground truth contact line. All axis are in [m] scale in world coordinate.

L2 Error [mm]	VIRDO++	w/o c_t	Rigid body asmp.
Contact Est.	6.647 (4.806)	9.872 (5.725)	21.203 (7.031)
Contact Pred.	7.198 (4.439)	9.356 (5.978)	22.572 (6.990)

Table. 2: Having latent embedding c_t as an input, VIRDO++ can decrease about 33 % of error for estimations and 23 % for predictions. Moreover, rigid body assumption is a worst case scenario producing $\times 3$ error.

Sec. 4.2, estimations are 1 [mm] more precise than predictions because they are optimized using the sensor measurements. Considering that the total length of the spatulas are about 200 [mm], this is less than 4% of error in total length. Estimating extrinsic contacts is a challenging task for rigid object [43] and is further complicated by object compliance; however, our approach proves effective due to its high-quality 3D reconstructions (details of error distribution in **Appendix** Fig. 12). Tab. 2 also quantifies performance against ablations (without c_t and Rigid-body assumption) illustrating the importance of the contact embedding and reasoning over compliance.

4.4 Generalizations Tasks

This section demonstrates VIRDO++ 's generalization ability to unseen objects and their dynamics. For each unseen object, the generalization dataset contains 1 nominal point cloud and 3 trajectories, each with 24 transitions with occlusion. Here, we directly apply the same pre-trained model from Sec. 4.1 to the generalization test dataset.

4.4.1 Object code inference for Unseen Objects

VIRDO++ is able to represent nominal geometries unseen during training within distribution. Fig. 6 shows reconstructions of both seen and unseen object in Fig. 3 using a single pretrained Object Module. For the unseen objects, we start from a randomly initialized object code $\alpha \sim \mathcal{N}(0, 0.01)$ and update the code by gradient descent with Adam. We use the same loss function from Eq. 1 with pre-trained VIRDO++ while freezing the weights of the network. The first row of Tab. 3 shows the Chamfer distance error of the unseen objects alongside training objects, indicating the relatively small drop in performance. When comparing Fig. 6 and the ground truth in Fig. 3, we note that the test reconstructions are close to the ground truth aside from some fine edge details.

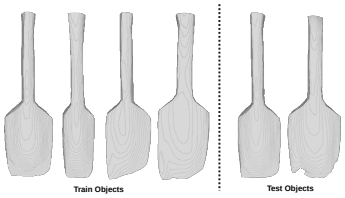

Fig. 6: VIRDO++ can reconstruct train and test objects with high fidelity (same order as Fig. 3).

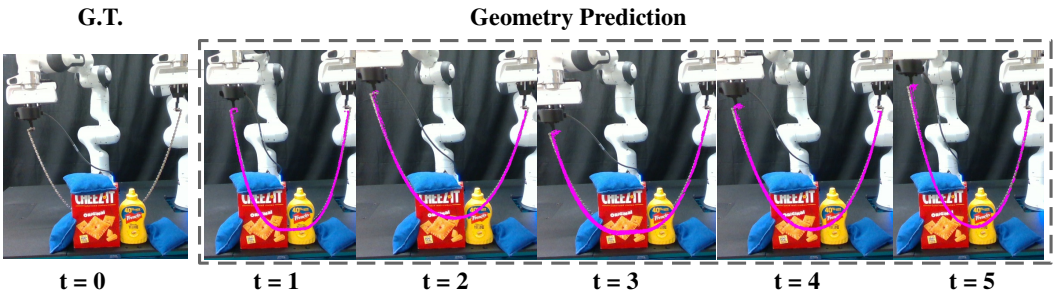

Fig. 7: VIRDO++ can accurately estimate the bike chain geometry (predictions overlaid in magenta) under occlusions due to objects at the front. Note that the chain intermittently makes contact and rests on the table, which is correctly predicted by VIRDO++ despite significant occlusion of where the chain makes contact.

4.4.2 Zero Shot Dynamics Prediction and State-estimation Generalization

With the object codes obtained in Sec. 4.4.1, VIRDO++ is able to predict the unseen object's dynamics based on their nominal shape. Here, we evaluate both prediction and state-estimation performance as in Sec. 4.2. We note that the observations of the unseen objects include occlusions as discussed in **Dataset**. Tab. 3 shows comparable performance between training and test objects with a small drop in accuracy, highlighting our model's generalization capability. Sec. A.3.6 in **Appendix** provides intuition for the Chamfer distance values. Further, we evaluate VIRDO++ 's ability to perform high-fidelity geometry prediction and estimation even in novel environment (Wok scraping) without 3D model. Details can be found in **Appendix** Sec. A.4.

CD ($\times 10^3$)	VIRDO++	w/o $\boldsymbol{c}_t$	Rigid Asm.	Test Obj. 1	Test Obj. 2
Geo. – Nom.	0.65 (0.08)	-	-	1.08	1.29
Geo. – S.E.	0.93 (0.21)	1.18 (0.35)	16.66 (11.51)	1.39 (0.20)	1.56 (0.29)
Geo. – Pred.	1.04 (0.35)	1.09 (0.25)	15.16 (10.57)	1.54 (0.45)	2.40 (1.08)
Wr. – Pred.	1.14 (1.05)	1.04 (1.01)	1.63 (0.89)	1.50 (1.00)	2.14 (1.75)

Table. 3: (Column 1-3) Ablation study using the train object's unseen trajectories. Numbers indicate mean (std.) of each error distribution. Chamfer distance used 5600 points and is multiplied by 10^3. (Column 4-5) Generalization results to unseen objects using the same pretrained VIRDO++ . Geo. = Geometry, Nom. = Nominal, S.E. = State Estimation, Pred. = Prediction, Wr. = Wrench calculated as $\|\hat{\boldsymbol{f}}_t - \boldsymbol{f}_t\|$.

4.5 Bike Chain Application

In this experiment, the chain ends are grasped by two arms. One end of the chain is moved randomly through the vertices of an equally-spaced $6 \times 6 \times 6$ grid fitting inside of a $0.1 \times 0.18 \times 0.2$ meter bounding box. The visited vertices of this grid make up the training dataset. The test dataset is generated by sampling 20 random positions uniformly from the bounding box. The end-effector orientations were fixed; therefore, $\boldsymbol{p}_t, \boldsymbol{a}_t \in \mathbb{R}^3$. As shown in Fig. 7, VIRDO++ is able to predict/estimate both free-space and in-contact chain geometry. The in-contact phase occurs when the arm moves sufficiently far down such that the chain lays on the table. We report the quantitative results of geometry estimation and dynamics prediction in Tab. 1 and **Appendix** Sec. A.3.3. The empirical results are comparable to spatula experiments despite significantly larger shape variations. Details on wrench prediction errors in **Appendix** Sec. A.3.4.

5 Limitations

Our proposed approach has a number of limitations: **1) Non-rigid grasps:** VIRDO++ assumes that the gripper is rigidly holding the object. When it comes to non-rigid grasping, we need to estimate relative motion of the objects w.r.t the gripper and the force transmission is now a function of this relative motion. One potential solution is to use collocated tactile sensor feedback [44, 45, 46] with F/T sensing to reason over reaction forces conditioned on the grasp. **2) Inertial effects:** our approach assumes quasi-static manipulation (negligible accelerations). During fast motions, the tools' inertial effects may also impact deformations and wrench measurements. When applicable, we suggest supplementing VIRDO++ 's input with end-effector accelerations. **3) Generalization to novel environments:** While we provide an example of environment generalization (**Appendix** Sec. A.4), the performance is lower than our other demonstrated applications. VIRDO++ may benefit from a 3D model of the environment to integrate inter-penetration losses [23] or collision checkers [47].

Acknowledgments

This work was in part supported by Robotics at Google under the Google Faculty Research Award 2021 and the National Science Foundation (NSF) grant NRI-2220876. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Bibliography

[1] Y. Wi, P. Florence, A. Zeng, and N. Fazeli. VIRDO: visio-tactile implicit representations of deformable objects. *CoRR*, abs/2202.00868, 2022. URL https://arxiv.org/abs/2202.00868.

[2] H. Yin, A. Varava, and D. Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54), 2021.

[3] 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. *The International Journal of Robotics Research*, 37(7):688–716, 2018.

[4] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt. Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, 7:82, 2020.

[5] Y. C. Hou, K. S. M. Sahari, and D. N. T. How. A review on modeling of flexible deformable object for dexterous robotic manipulation. *International Journal of Advanced Robotic Systems*, 16(3): 1729881419848894, 2019.

[6] I. A. Essa, S. Sclaroff, and A. Pentland. A unified approach for physical and geometric modeling for graphics and animation. In *Computer Graphics Forum*, volume 11, pages 129–138. Wiley Online Library, 1992.

[7] F. Ficuciello, A. Migliozzi, E. Coevoet, A. Petit, and C. Duriez. Fem-based deformation control for dexterous manipulation of 3d soft objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4007–4013. IEEE, 2018.

[8] B. Frank, C. Stachniss, R. Schmedding, M. Teschner, and W. Burgard. Learning object deformation models for robot motion planning. *Robotics and Autonomous Systems*, 62(8):1153–1174, 2014.

[9] M. Müller and M. H. Gross. Interactive virtual materials. In *Graphics interface*, volume 2004, pages 239–246, 2004.

[10] A. Sengupta, A. Krupa, and E. Marchand. Tracking of non-rigid objects using rgb-d camera. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3310–3317. IEEE, 2019.

[11] A. M. Howard and G. A. Bekey. Intelligent learning for deformable object manipulation. *Autonomous Robots*, 9(1):51–58, 2000.

[12] A. Nurnberger, A. Radetzky, and R. Kruse. A problem specific recurrent neural network for the description and simulation of dynamic spring models. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, pages 468–473. IEEE, 1998.

[13] D. L. Tonnesen. *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation.* Citeseer, 2000.

[14] B. Thach, B. Y. Cho, A. Kuntz, and T. Hermans. Learning visual shape control of novel 3d deformable objects from partial-view point clouds. *arXiv preprint arXiv:2110.04685*, 2021.

[15] B. Thach, A. Kuntz, and T. Hermans. Deformernet: A deep learning approach to 3d deformable object manipulation. *arXiv preprint arXiv:2107.08067*, 2021.

[16] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, pages 192–202. PMLR, 2022.

[17] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. *arXiv preprint arXiv:2203.17275*, 2022.

[18] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg. Deep transfer learning of pick points on fabric for robot bed-making. *arXiv preprint arXiv:1809.09810*, 2018.

[19] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. K. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg. Visuospatial foresight for physical sequential fabric manipulation. *Autonomous Robots*, pages 1–25, 2021.

[20] P. Sundaresan, R. Antonova, and J. Bohg. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. *arXiv preprint arXiv:2204.03139*, 2022.

[21] R. Antonova, J. Yang, P. Sundaresan, D. Fox, F. Ramos, and J. Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3):5819–5826, 2022. doi:10.1109/LRA.2022.3157377.

[22] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, et al. Factory: Fast contact for robotic assembly. *arXiv preprint arXiv:2205.03532*, 2022.

[23] A. Petit, S. Cotin, V. Lippiello, and B. Siciliano. Capturing deformations of interacting non-rigid objects using rgb-d data. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 491–497. IEEE, 2018.

[24] J. Zhang, Y. Zhong, J. Smith, and C. Gu. Neural dynamics-based poisson propagation for deformable modelling. *Neural Computing and Applications*, 31(2):1091–1101, 2019.

[25] Y. Wang, D. McConachie, and D. Berenson. Tracking partially-occluded deformable objects while enforcing geometric constraints. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14199–14205. IEEE, 2021.

[26] P. Palafox, A. Božič, J. Thies, M. Nießner, and A. Dai. Npms: Neural parametric models for 3d deformable shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12695–12705, October 2021.

[27] X. Ma, D. Hsu, and W. S. Lee. Learning latent graph dynamics for deformable object manipulation. *arXiv preprint arXiv:2104.12149*, 2021.

[28] I. Santesteban, N. Thuerey, M. A. Otaduy, and D. Casas. Self-supervised collision handling via generative 3d garment models for virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11763–11773, June 2021.

[29] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. *arXiv preprint arXiv:2205.02909*, 2022.

[30] Z. Huang, X. Lin, and D. Held. Mesh-based dynamics with occlusion reasoning for cloth manipulation. *arXiv preprint arXiv:2206.02881*, 2022.

[31] C. Chi and S. Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. In *The IEEE International Conference on Computer Vision (ICCV)*, 2021.

[32] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022.

[33] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *International Conference on Computer Vision*, Oct. 2019.

[34] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[35] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020.

[36] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[37] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[38] Y. Deng, J. Yang, and X. Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. *CoRR*, abs/2011.13650, 2020. URL https://arxiv.org/abs/2011.13650.

[39] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[40] D. Ma, S. Dong, and A. Rodriguez. Extrinsic contact sensing with relative-motion tracking from distributed tactile measurements. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11262–11268, 2021. doi:10.1109/ICRA48506.2021.9561781.

[41] Q. Li, C. SchÃ¼rmann, R. Haschke, and H. Ritter. A control framework for tactile servoing. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. doi:10.15607/RSS.2013.IX.045.

[42] M. Van der Merwe, D. Berenson, and N. Fazeli. Learning the dynamics of compliant tool-environment interaction for visuo-tactile contact servoing, 2022.

[43] A. Sipos and N. Fazeli. Simultaneous contact location and object pose estimation using proprioceptive tactile feedback. *arXiv preprint arXiv:2206.01245*, 2022.

[44] I. Taylor, S. Dong, and A. Rodriguez. Gelslim3. 0: High-resolution measurement of shape, force and slip in a compact tactile-sensing finger. *arXiv preprint arXiv:2103.12269*, 2021.

[45] J. Yin, G. M. Campbell, J. Pikul, and M. Yim. Multimodal proximity and visuotactile sensing with a selectively transmissive soft membrane. In *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pages 802–808. IEEE, 2022.

[46] N. Kuppuswamy, A. Alspach, A. Uttamchandani, S. Creasey, T. Ikeda, and R. Tedrake. Soft-bubble grippers for robust and perceptive manipulation. *arXiv preprint arXiv:2004.03691*, 2020.

[47] P. Mitrano, D. McConachie, and D. Berenson. Learning where to trust unreliable models in an unstructured world for deformable object manipulation. *Science Robotics*, 6(54):eabd8170, 2021.

[48] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun. Grnet: Gridding residual network for dense point cloud completion. In *European Conference on Computer Vision*, pages 365–381. Springer, 2020.

[49] J. Schauer and A. Nüchter. Collision detection between point clouds using an efficient kd tree implementation. *Advanced Engineering Informatics*, 29(3):440–458, 2015.

[50] J. Klein and G. Zachmann. Point cloud collision detection. In *Computer Graphics Forum*, volume 23, pages 567–576. Wiley Online Library, 2004.

[51] R. Weller. A brief overview of collision detection. *New Geometric Data Structures for Collision Detection and Haptics*, pages 9–46, 2013.

[52] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

A Appendix

Contents

A.1 Training Details . . . 12
A.2 Inference Details . . . 12
A.2.1 Parameters & Hardware . . . 12
A.2.2 Inference Algorithm Variations . . . 12
A.3 Experiment Details . . . 13
A.3.1 Differences in Physical Properties of Spatulas . . . 13
A.3.2 Coordinate System and Action Distribution . . . 13
A.3.3 Distribution of Geometry and Wrench State Estimation and Predictions . . . 13
A.3.4 Reaction Wrench Prediction Errors . . . 14
A.3.5 Details o Contact Detection Errors . . . 15
A.3.6 Details on Generalization to Novel Objects Dynamics . . . 15
A.4 Generalization to Novel Environments . . . 15
A.5 Ablation study on design choices . . . 16
A.5.1 Explicit Geometry Representation . . . 16
A.5.2 Implicit Geometry Representation with Different Model Architectures . . . 16

A.1 Training Details

Tab. 4 contains training parameters. In addition, we used 2,000 on-surface points and 20,000 off-surfaces for training VIRDO++ . We used the Adam optimizer with learning rate = 10^{-4} and epoch = 200.

l_c	λ	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}	λ_{11}
6	5e1	1e1	1e1	1e4	1	1e2	1e5	5e1	3e6	1e1	1e1	1e1

Table. 4: Parameters used for training VIRDO++

A.2 Inference Details

A.2.1 Parameters & Hardware

For the inference, we use number of particles $n = 40$, resampling parameter $\gamma = 0.7$, and epoch $e = 10$. This results in inference time = 0.7 [s] with GPU = NVIDIA A6000 and CPU= 32-Core 3.50 GHz AMD. We note that one can balance the state-estimation accuracy between the inference time by changing the number of particles and number of epochs. Here, we note that the epochs need not to be large due to Alg. 1 which provides a close-to-ground-truth initial guess on the object's state.

A.2.2 Inference Algorithm Variations

Algorithm 2 Resampling Algorithm

function RE-SAMPLING($\boldsymbol{w}_t, \mathbf{C}_t$)
  Draw $\{\bar{\boldsymbol{C}}_t^0, \bar{\boldsymbol{C}}_t^1, ..., \bar{\boldsymbol{C}}_t^{k-1}\} \sim \mathcal{N}(0, 0.01)$ ▷ Exploration
  Draw $\{\bar{\boldsymbol{C}}_t^k, \bar{\boldsymbol{C}}_t^{k+1}, ..., \bar{\boldsymbol{C}}_t^{N-1}\} \sim \{(\boldsymbol{C}_t^0, w_t^0), (\boldsymbol{C}_t^1, w_t^1)..., (\boldsymbol{C}_t^{N-1}, w_t^{N-1})\}$ ▷ Exploitation
  return $\mathbf{C}_t$
end function

The input of the resampling function in Alg. 1 is the beliefs over the current states ($\mathbf{C}_t$) and the corresponding reliability, a.k.a the sampling weight ($\boldsymbol{w}_t$). The resampling function refines $\mathbf{C}_t$ to $\bar{\mathbf{C}}_t$ through Alg. 2. In the

CD ($x10^3$)	$\beta = 0\%$	$\beta = 25\%$	$\beta = 50\%$	$\beta = 75\%$
Geom. – S.E.	0.936 (0.206)	0.937 (0.209)	0.941 (0.236)	0.943 (0.202)
Geom. – Pred.	1.035 (0.335)	1.046 (0.358)	1.049 (0.370)	1.117 (0.497)
Force – Pred.	1.241 (1.282))	1.125 (1.068)	1.149 (1.069)	1.326 (1.282)
Torque – Pred.	0.140 (0.109)	0.134 (0.099)	0.134 (0.095)	0.158 (0.111)

Table. 5: $\beta = 0 \sim 50\%$ does not significantly increase nor decrease the performance of our Inference algorithm. However, when exploration becomes dominant over the exploitation ($\beta = 75\%$), a distinguishable drop in performance is observed.

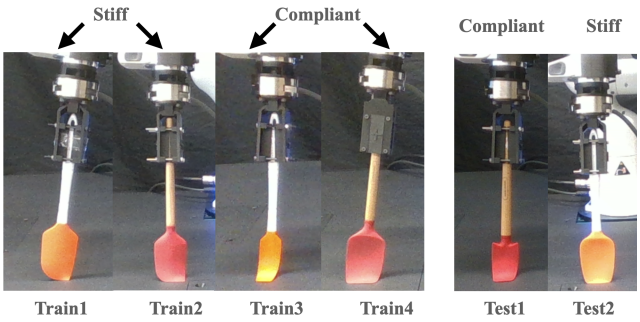

Fig. 8: We maintained the pressing force of 6 [N] for all six spatulas and measured the difference in deformations. The combination of material property and geometry determines stiffness/compliance. The spatulas with the same tip color consist of the same material; however, they deform very differently depending on their structures.

exploitation step, we compose k particles in $\bar{\mathbf{C}}_t$ by sampling from $\sim \mathcal{N}(0, 0.01)$. In the exploration step, we sample the rest of $\bar{\mathbf{C}}_t$ by sampling from $\boldsymbol{w}_t$.

Our ablation study (presented in Tab. 5) aims to find an optimal balance between exploration and exploitation. Here, the "Force-Prediction" is calculated as $\|\hat{\boldsymbol{f}}_t[:\ 3] - \boldsymbol{f}_t[:\ 3]\|$ and "Torque-Prediction" is calculated as $\|\hat{\boldsymbol{f}}_t[3\ :] - \boldsymbol{f}_t[3\ :]\|$. We found that $\beta = 0$ and 25% has comparable state estimation results. Geometry prediction is the most accurate when $\beta = 0\%$; however, wrench predictions are most accurate when $\beta = 25\%$. When compared to $\beta = 0\%$, $\beta = 25\%$ has a 10% increased geometry prediction error and $9 - 10\%$ reduced wrench prediction error.

A.3 Experiment Details

A.3.1 Differences in Physical Properties of Spatulas

In this section, we illustrate how the train/test spatulas deform relative to each other when brought into contact with the table in a similar configurations (affected by the arm impedance controller) and same normal force (here, 6 [N] along the z axis). Fig. 8 shows a wide distribution of deformations achieved within the same object category. With the same force, train object 1 barely deforms and is close to it's nominal shape, whereas train object 3 bends significantly. This difference in physical properties makes generalization very difficult, even within the same object class. For example, test object 2 visually resembles train object 4; however, it has stiffness properties more similar to train object 2. This has lead to VIRDO++ producing higher errors in the generalization task for test 2 objects.

A.3.2 Coordinate System and Action Distribution

Pointcloud are collected with respect to a reference frame defined at the end of the F/T sensor mounted at the robot's wrist. Actions are the Cartesian motion of the end-effector of the robot with respect to the base frame, Fig. 9. Fig. 10 visualizes action distributions of train and test trajectories of training objects. Both are normally distributed and share mean and standard deviation because they were generated from the same action policy.

A.3.3 Distribution of Geometry and Wrench State Estimation and Predictions

Fig. 11 shows detailed distribution of errors reported in Tab. 1. The Kernel Density Estimate (KDE) plot shows that the errors are normally distributed. The training error mean is smaller than the test error for all graphs

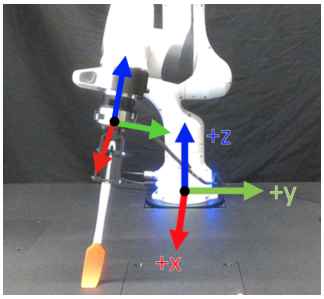

Fig. 9: Action frame is defined at the robot's base frame while pointclouds are registered with respect to the end-effector frame.

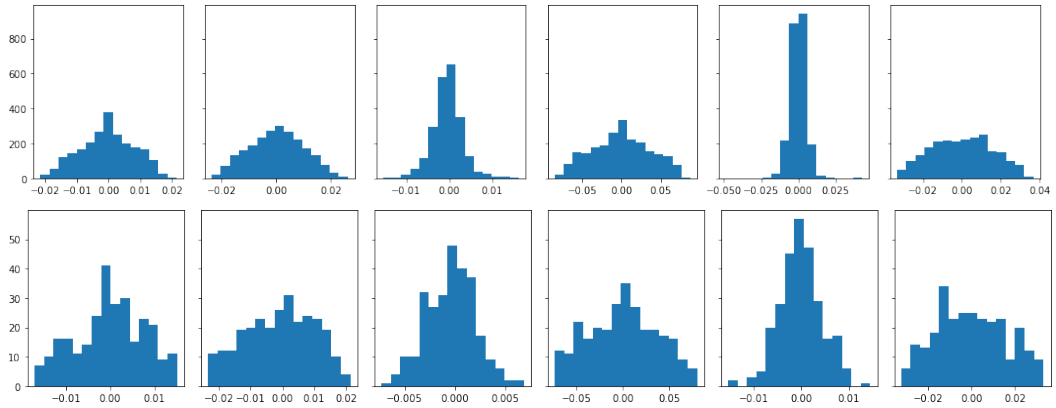

Fig. 10: Distributions of training and testing action trajectories. These distributions are intentionally centered around zero. Top row = Train, Bottom row = Test. Fx, Fy, Fz, Tx, Ty, Tz order where x-axis is either force[N] or Torque[Nm]. y-axis quantifies the number of samples.

in both demos. In addition, geometry estimation is more accurate than geometry predictions as it uses sensor measurements. For example, in the spatula demo, the Chamfer distance error at the peak geometry estimation from the training set (green dashed line) is smaller than the test set (solid dashed line) and is also smaller than the prediction estimates (yellow dashed line).

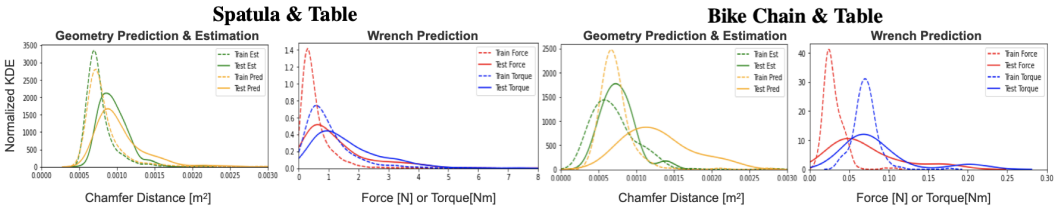

Fig. 11: Kernel Density Estimate (KDE) plots for spatula and bike chain applications show that accro's geometry and wrench predictions and state-estimation errors are within small margins. Force prediction errors = $\|\hat{\boldsymbol{f}}_t[:3] - \boldsymbol{f}_t[:3]\|$ [N]. Torque prediction error as $10 * \|\hat{\boldsymbol{f}}_t[3:] - \boldsymbol{f}_t[3:]\|$ [Nm].

A.3.4 Reaction Wrench Prediction Errors

Tab. 6 shows the wrench predictions results of trained objects (Tab. 3's first column). Because we use L2 norm for reaction wrench prediction during training, error distributions are centered around 0, and the degree of errors appears as standard deviation.

The standard deviation in error for the chain task is approximately an order of magnitude smaller than the spatula. We hypothesize that this because i) the ground truth wrench is an order of magnitude smaller than the spatula and ii) the action space dimension is half the spatula resulting in a simpler model to learn.

l_1 Error	f_x [N]	f_y [N]	f_z [N]	τ_x [Nm]	τ_y [Nm]	τ_z [Nm]
Sp. Train	0.01 (0.31)	-0.00 (0.36)	0.03 (0.65)	-0.00 (0.11)	0.00 (0.09)	0.00 (0.02)
Sp. Test	-0.03 (0.35)	0.08 (0.54)	-0.28 (1.60)	-0.02 (0.16)	-0.01 (0.11)	-0.00 (0.02)
Ch. Train	0.02 (0.01)	0.01 (0.01)	0.02 (0.01)	-0.01 (0.00)	0.00 (0.01)	0.01 (0.00)
Ch. Test	0.02 (0.06)	0.01 (0.03)	0.03 (0.01)	-0.00 (0.00)	0.00 (0.01)	0.00 (0.00)

Table. 6: **Mean and Std Wrench Prediction Errors:** $\hat{\boldsymbol{f}}_t - \boldsymbol{f}_t$ for the spatula and chain experiments.

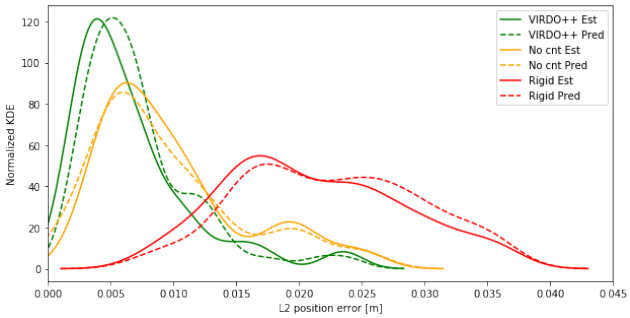

Fig. 12: VIRDO++ 's contact detection outperforms the baselines without c_t and rigid body assumption where the detailed quantitative analysis is in Tab. 2

A.3.5 Details o Contact Detection Errors

Fig. 12 contains detailed error distributions of Tab. 2 from the main text. VIRDO++ (green)'s Chamfer distance has a significantly smaller mean and standard deviation compared to the baseline models without contact embedding and compliance assumption. One interesting observation is that the geometry prediction (yellow dashed line) is located left to the geometry estimation (solid yellow line). This is because the model without c_t produces deterministic reconstructions as a function of p_t and f_t, leaving no room for improving state estimation with the Bayesian filtering algorithm.

A.3.6 Details on Generalization to Novel Objects Dynamics

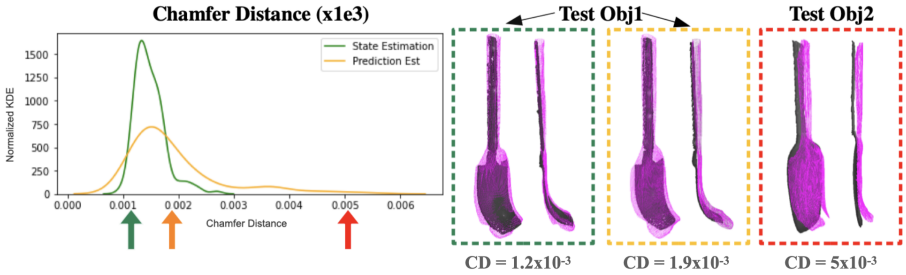

Fig. 13: **Generalizing to unseen object dynamics**. Left panel: KDE plot of Chamfer distances (x1e3) $[m^2]$ using normalized geometries. We show both test objects state estimation using partial point clouds and geometry predictions. Similar to training objects, state estimation has smaller mean and standard deviation of the error. Right panel: We visualized three representative generalization results and corresponding Chamfer distances.

Fig. 13 is a detailed Chamfer distance distribution of Tab. 3 from the main text and the visualization of each Chamfer distance loss. The purpose of reconstructions visualized in Fig. 13 (right) is to further provide an intuition of each error and how to interpret its magnitude. For example, the Chamfer distance value of 0.002 for the unseen objects is about double the amount of the training objects' average state estimation error; however, in visualization, the reconstructions can reasonably predict the ground truth's curvature. We observed that the majority of errors are due to small undesirable artifacts at the edges of objects which do not significantly impact the general trend of deformations.

A.4 Generalization to Novel Environments

In this section, we show VIRDO++ generalization capability to novel environments. Here, the novel environment is a wok scraping task. The wok presents high curvatures producing greater variances in contact formations with respect to the flat table. Here, we add a training dataset with 2 scraping trajectories on the wok and perform state estimation and predictions given a test trajectory with 13 transitions. As in Fig. 14, VIRDO++ provides high-fidelity dense geometry of deformable objects in high occlusion cases without a 3D

model or configuration of the environment; however, due to the complex environment, it occasionally produces undesired artifacts (2 out of 13 trans.).

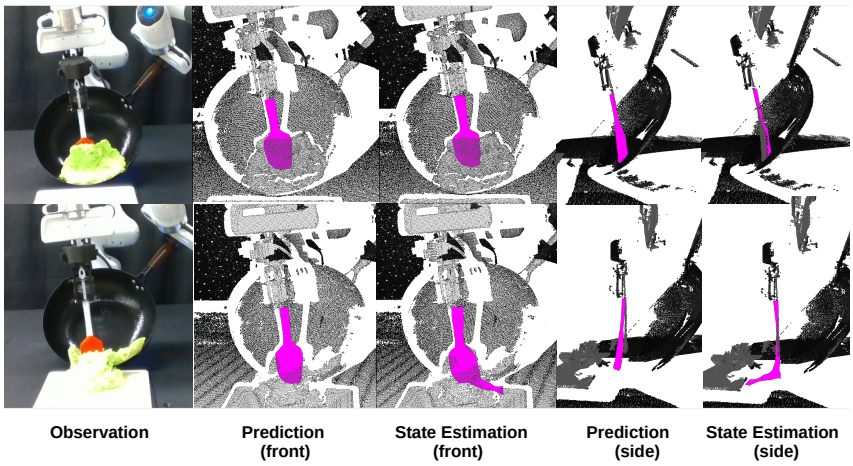

Fig. 14: VIRDO++ can reconstruct high-fidelty dense geometry of deformable objects under occlusions in new environments, such as scraping objects off a wok (successes, top row), but can also at times produce undesirable artifacts (failure modes shown in bottom row).

A.5 Ablation study on design choices

A.5.1 Explicit Geometry Representation

In this section, we compare VIRDO++ with explicit models by replacing the geometry representations (deformation and object module) with GRNet's pointcloud encoder-decoder structure [48]. Here, we use the same force module as VIRDO++ because the baseline does not have a multi-modal sensory encoder. The explicit dynamics model directly takes in a partial pointcloud at time t to generate a dense pointcloud at time $t+1$. The inputs to the dynamics model are pointcloud at the current time step P_t, force code z_t, contact latent vectorc_t, and action a_t and the output is full pointcloud at the next time step P_{t+1}. While P_t is directly input to [48]'s encoder, and other inputs are concatenated to the bottle-neck features.

Here, we used the same dataset in Sec. 4.1. For the loss function, we replaced $\mathcal{L}_t^{geo}$ to Chamfer distance as [48] proposed. The second column of Tab. 7 shows the result of the explicit dynamics model based on [48], where the learned explicit model showed better result than VIRDO++ for training trajectories. However, the performance dropped significantly given the pointcloud with occlusions from the test trajectories. We speculate that because the explicit dynamics model directly consumes the partial pointcloud to roll out the dynamics, it is more susceptible to visual outliers caused by occlusions. However, VIRDO++ is more robust to occlusions due to its implicit nature, using partial pointcloud implicitely via solving the optimization problem in Sec. 3.4

Besides dealing with occlusions, VIRDO++ 's implicit dynamics model is also advantageous over the explicit counterparts in multi-object interactions. Specifically, the detection of contact and collision at a query point can easily be done by evaluating the sign-distances from each object's signed distance field at a time complexity of $\mathcal{O}(number\ of\ objects)$. On the other hand, explicit geometric representations require special contact/collision detection techniques such as k-d tree [49] or sphere coverings [50]. These methods have higher time complexities dependent on the length of each point cloud. [51, 50, 49]. In Sec.4.3, we demonstrate the accurateness of VIRDO++ for detecting contact location.

A.5.2 Implicit Geometry Representation with Different Model Architectures

In this section, we evaluate variations of VIRDO's geometry representation using DeepSDF [37], SIREN [36], and a positional encoder [52]. In this ablation study we first attempted to replace the VIRDO++ 's entire geometry representation modules (i.e., both deformation and shape module) with [37], [36], and [52], then train each model end to end with the VIRDO++ 's force and action modules. Under this condition, we set the input as $(\boldsymbol{x}, \alpha, z_t)$ and the output as a signed distance function$s \in \mathbb{R}$, where $\boldsymbol{x}$ is the query point instead of the entire pointcloud as Sec. A.5.1. However, training end to end without residual layers resulted in failure of convergence for all variations.

We observed that fixing the weights of the pre-trained object modules and learning only the residuals (deformations) significantly helped the models converge. We chose to replace only the deformation module with the proposed variations and compare performance. The inputs are $(\boldsymbol{x}, \alpha, z_t)$ and the output is $\Delta\boldsymbol{x}$. We used the

CD ($\times 10^3$)	VIRDO++	GRNet	DeepSDF	SIREN	Pos. Enc.
Train Est.	0.777 (0.215)	–	5.087 (0.737)	5.521 (1.015)	5.569 (2.847)
Train Pred.	0.830 (0.330)	0.759 (0.667)	5.667 (1.306)	5.457 (0.888)	5.886 (2.561)
Test Est.	0.934 (0.210)	–	10.016 (3.253)	6.349 (1.388)	8.081 (3.790)
Test Pred.	1.041 (0.348)	1.230 (0.826)	9.552 (2.767)	6.988 (1.280)	7.188 (3.603)

Table. 7: Results are evaluated in the real-world, showing that VIRDO++ outperforms its implicit and explicit counterparts.

same dataset and loss functions as VIRDO++ except for training DeepSDF, where we removed $\mathcal{L}_{hyper}$ when training DeepSDF as the architecture does not have a hyper-network.

Template-based DeepSDF: For an apple-to-apple comparison, we set the number of hidden layers = 2 and hidden features = 256 for DeepSDF structures to the same size in VIRDO++ 's deformation module. We observe that Chamfer distance error becomes $6.54 \sim 6.82$ time greater for the training trajectories and $9.18 \sim 10.72$ times greater for the unseen trajectories when changing the hypernetwork to the DeepSDF architecture. The hypernetwork structure has higher representation power, resulting in more precise geometry reconstructions.

SIREN: Here, we replaced the geometry representation with SIREN's hypernetwork and trained it with the force and action modules. As suggested in the paper, we implemented SIREN with 5 layers and hidden features of size 256. HyperNetwork (ReLU MLP) was initialized with Kaiming initialization, and the HypoNetwork was initialized as $w_i \sim \mathcal{U}(-1/l, +1/l)$, where w_i is the network's weight and l is the total number of network parameters. We found that the sine activation function generates bumpy textures resulting in high reconstruction errors for geometry estimation and predictions.

Positional Encoder: For this experiment, we add a positional encoder from [52] right before VIRDO++ 's deformation modules. Similar to what we observed from SIREN, adding sine or cosine non-linearity to the network can produce undesirable artifacts. We found that the positional encoder generate floating blobs in the free space, resulting in much higher errors than the original VIRDO++ .