Abstract—Manipulation planning is the problem of finding a sequence of robot configurations that involves interactions with objects in the scene, e.g., grasping and placing an object, or more general tool-use. To achieve such interactions, traditional approaches require hand-engineering of object representations and interaction constraints, which easily becomes tedious when complex objects/interactions are considered. Inspired by recent advances in 3D modeling, e.g., NeRF, we propose a method to represent objects as continuous functions upon which constraint features are defined and jointly trained. In particular, the proposed pixel-aligned representation is directly inferred from images with known camera geometry and naturally acts as a perception component in the whole manipulation pipeline, thereby enabling long-horizon planning only from visual input.

Index Terms—Integrated planning and learning, manipulation planning, representation learning.

I. INTRODUCTION

Dexterous robots should be able to flexibly interact with objects in the environment, such as grasping and placing an object, or more general tool-use, to achieve a certain goal. Such instances are formalized as manipulation planning, a type of motion planning problem that solves not only for the robot’s own movement but also for the objects’ motions subject to their interaction constraints. Therefore, designing interaction constraint functions, which we also call interaction features, is at the core of achieving the robot dexterity. Traditional approaches rely on hand-crafted constraint functions based on geometric object representations such as meshes or combinations of shape primitives. However, when considering large varieties of objects and interaction modes, such traditional approaches have long-standing limitations in two aspects: i) The representations have to be inferred from raw sensory inputs like images or point clouds – raising the fundamental problem of perception and shape estimation. ii) With increasing generality of object shapes and interaction, representation’s complexity grows, thereby making hand-engineering of the interaction features inefficient. However, if the aim is manipulation skills, the hard problem of precise shape estimation and the feature engineering might be unnecessary.

What is a good object representation? Considering the representation will be used to predict interaction features, we expect it to encode primarily task-specific information rather than only geometric. We also expect some of the information to be shared across different interaction modes. In other words, good representations should be task-specific so that the feature prediction can be simplified and, at the same time, be task-agnostic to enable synergies between the tasks. E.g., mug handles are called handles because we can handle the mug through them and also, once we learn the notion of a handle, we can play around with the mug through the handle in many different ways. Also, from the perception standpoint, good representations should be easy to infer from raw sensory inputs and should be able to trade their accuracy (if bounded) in favor of the feature prediction.

To this end, we propose a data-driven approach to learning interaction features that are conditioned on object images. The whole pipeline is trained end-to-end directly with the task supervisions so as to make the representation and perception task-specific and thus to simplify the interaction prediction. The object representation acts as a bottleneck and is shared across multiple features so that the task-agnostic aspects can emerge. We propose the representation to be a $d$-dimensional continuous function over the 3D space [1], [2]. In particular, the proposed implicit neural representation is pixel-aligned, meaning that the function takes as input images from multiple cameras (e.g. stereo) and, assuming known camera poses and intrinsics, computes a representation at a certain 3D location using image features at the corresponding 2D pixel coordinates. Once learned, the interaction features can be used by a typical constrained optimal control framework to plan dexterous object-robot interaction. We show that making use of the learned constraint models within Logic-Geometric Programming (LGP) [3] enables planning various types of interactions with complex-shaped objects only from images. Since the representations generalize well, the learned constraint models are directly applicable to manipulation tasks involving unseen objects.

\[1\]

While this letter version has been made self-contained, the full text, which includes extensive literature reviews and technical/experimental details, videos and a pytorch implementation can be found on our project page: https://sites.google.com/view/deep-visual-constraints
II. RELATED WORK

Implicit neural representations have recently gained increasing attention in 3D modeling. The core idea is to encode an object or a scene in the weights of a neural network, where the network acts as a direct mapping from 3D spatial location to an implicit representation of the model, such as occupancy measures [4], signed distance fields (SDF) [1], [5], or radiance fields [2]. In contrast to explicit representations like voxels, meshes or point clouds, the implicit representations don’t require discretization of the 3D space nor fixed shape topology but rather continuously represent the 3D geometry, thereby allowing for capturing complex shape geometry at high resolutions in a memory efficient way. The idea has made particularly great successes in the application of view synthesis [2], [6], creating photo-realistic images from novel views. Beyond the shape/appearance modeling, our work makes use of implicit neural representation to model physical interaction feasibility and thereby to provide a differentiable constraint model for robot manipulation planning.

Robotics community has also attempted to adopt the idea of implicit neural representations. Albeit 2D, [7], [8] trained fully-convolutional neural networks (FCNs) that map a raw input image to pixel-wise dense object representation which directly generalizes to unseen objects. Similarly, [9], [10] used dense pixel-wise object/scene descriptions to address manipulation scenarios such as throwing or pick-and-place, where FCNs predicted the task score maps. In addition, [11], [12] proposed 3D FCNs that take as input a truncated SDF and return the grasp affordance fields. In [13], the PointNet encoder maps point clouds into 3D dense object representation which is then used for motion imitations from few demonstrations. [14] formulated manipulation planning problems solely in terms of SDFs as representations and proposed to learn manipulation constraints as functionals of SDFs. More recently, [15] trained implicit object encoders together with differentiable image rendering decoders and used a graph neural network to model dynamics, based on which an RRT-based method can plan sequential manipulations in the latent space. In contrast to the above, our proposed representations are trained in conjunction with multiple task prediction heads and can be seamlessly integrated into sequential manipulation planning schemes that generate motions flexibly blending diverse interactions together.

III. DEEP VISUAL CONSTRAINTS (DVC)

Given $N_{\text{view}}$ images with their camera poses/intrinsics, $\mathcal{V} = \{(I_1, T_1, K_1), \ldots , (I_{N_{\text{view}}}, T_{N_{\text{view}}}, K_{N_{\text{view}}})\}$ with $I \in \mathbb{R}^{3 \times H \times W}$ (we considered $H = W = 128$) and $T, K \in \mathbb{R}^{4 \times 4}$, we build an interaction feature as a neural network:

$$h = \phi_{\text{task}}(q; \mathcal{V}),$$

where $q \in SE(3)$ is the pose of the robot/static frame interacting with the object; the interaction feature $h \in \mathbb{R}$, analogous to energy potentials, is zero when feasible and non-zero otherwise, which will act as an equality constraint in manipulation planning. As shown in Fig. 2, the feature prediction framework consists of two parts: the representation backbone which serves as an implicit representation of an object, and the task heads that make feature predictions. Notably, while the multiple task heads individually model different interaction constraints, the backbone is shared across them, allowing for learning more general object representation.

A. Pixel-Aligned Implicit Functional Object (PIFO)

The proposed implicit object representation is a mapping:

$$y = \psi(p; \mathcal{V}),$$

where $p \in \mathbb{R}^3$ and $y \in \mathbb{R}^d$ are a queried 3D position and a representation vector at that point, respectively. This function, implemented as a neural network as depicted in Fig. 3, consists
of three parts: image encoder, 3D reprojector, and feature aggregator. The first two compute a representation vector from each image and the last one combines them.

**Image Encoder:** This module takes as input an image and computes a feature map (the pathway from $I^n$ to $F^n$ in Fig. 3). We adopted the hourglass network architecture, especially with ResNet-34 as its downward path and two residual layers with $3 \times 3$ convolutions followed by up-convolution as the upward path:

$$F^n = UNet(I^n), \forall n \in \{1, \ldots, N_{\text{view}}\},$$

which results in a feature map $F^n \in \mathbb{R}^{64 \times 64 \times 64}$ that captures both local and global information in the input image.

**3D Reprojector:** To endow the network with the multi-view consistency, all the 3D operations are performed in the view space. The 3D reprojector, the pathway from $(T^n, K^n)$ and $p$ to $y^n$ in Fig. 3, transforms a queried point, $p$, into the image coordinate including depth, $\pi(p; T^n, K^n) = z \in \mathbb{R}^3$ and extracts the local image feature at the projected point from the feature map, $F^n$, via bilinear interpolation. Finally, the extracted feature and the coordinate feature, which is computed through a couple of fully connected layers (FCLs), are passed to a couple of FCLs to get a representation vector at $p$ for a single image, i.e., $\forall n \in \{1, \ldots, N_{\text{view}}\}$,

$$y^n = MLP(F^n(z^n), z^n), z^n = \pi(p; T^n, K^n).$$

**Feature Aggregator:** This module is the pathway from $y^n$ to $y$ in Fig. 3, which aggregates the representation vectors from multiple views into one vector. Among many permutation-invariant options, like summation or more sophisticated attention mechanisms, we simply take the averaging operation for it, i.e., $y = \frac{1}{N_{\text{view}}} \sum_{n=1}^{N_{\text{view}}} y^n$.

**B. Interaction Task Feature Prediction**

A task head evaluates the interaction constraint violation, $h$, for a given robot/static frame’s pose, $q$, using the object representation function over 3D, $\psi(\cdot)$. To this end, we rigidly attach a set of keypoints to the robot frame at which the backbone is queried, i.e., $\forall k \in \{1, \ldots, K\}$, $y_k = \psi(p_k; \mathcal{V}), p_k = R(q)\hat{p}_k + t(q)$, where $\hat{p}_k$ is the $k$th keypoint’s local coordinate, and $R(q)$ and $t(q)$ denote the rotation matrix and the translation vector of $q$, respectively. Finally, the task head, based on the resulting representation vectors, predicts a constraint value through a couple of FCLs:

$$h = MLP(y_1, \ldots, y_K).$$

### IV. Training

In this letter, we consider manipulation scenarios where a robot arm, Franka Emika Panda, or two manipulate mugs. The shapes of mugs are diverse and the scene contains multiple hooks on which a mug can be hung. Formulating such problems requires three types of learned interaction features: an SDF feature for collision avoidance and grasping/hanging features, so we prepared the dataset for each.

**A. Data Generation**

We took 131 mesh models of mugs from ShapeNet [16] and convex-decomposed those meshes. The meshes are translated and randomly scaled so that they can fit in a bounding sphere with a radius of $10 \sim 15$ cm at the origin. For each mug, we created the following dataset.

**Posed Images:** The posed image data consists of 100 images (128 x 128) with the corresponding camera poses and intrinsic matrices generated by the OpenGL rendering. Azimuths and elevations of the cameras are sampled such that they are uniformly projected onto the unit sphere, while their distances from the object center are random. The azimuth, elevation and distance fully determine the camera’s positions, and the camera’s orientations are set such that the cameras are upright and face the object center. For the intrinsics, we used the field of view $f_{\text{ov}} = 2 \arcsin(d/r)$, where $d$ is the camera distance from the object center and $r$ is the radius of the object’s bounding sphere, so that the object spans the entire image. Lighting is also randomized.

**SDF:** We sampled 12,500 3D points and precomputed their signed distance values, i.e., the distance of a point from the object surface with the sign indicating whether or not the point is inside the surface. Following the approach of DeepSDF [1],
we sampled more aggressively near the object surface to foster the learning of the object geometry.

Grasping & Hanging: The grasping and hanging data are 1,000 feasible grasping and hanging poses of the gripper and the hook, respectively. For grasping, we used an antipodal sampling scheme, similarly to [17], to create candidate gripper poses and checked their feasibility using Bullet [18]. For hanging, we randomly sampled collision-free hook poses and checked if it’s kinematically trapped by the mug in the directions perpendicular to the hook’s main axis.

In the end, we have a dataset of:
\[
\{(\hat{T}^{1:100}, \hat{T}^{1:100}, \hat{K}^{1:100}, p^{1:12500}), SDF^{1:12500}, q_{\text{grasp}}^{1:1000}, q_{\text{hang}}^{1:1000}\}^{131}_{i=1},
\]
which we divided into 78 train, 25 validation, 28 test sets.

B. Data Augmentation

While randomizing the azimuth, elevation and distance of the camera provides all possible appearances of the object, it still cannot account for varying roll angles of the camera (i.e. image rotations) and off-centered images. To show the network all possible images that it can encounter when deployed later and to mitigate the size-ambiguity issue, we propose to use a data augmentation technique based on Homography warping: In each iteration, for a randomly sampled set of images, we artificially perturb the roll angle of each camera and the estimated object center position (at which the cameras are looking). Also, fov is modified as if the radius of the bounding sphere is 15 cm so that smaller objects can appear smaller in the transformed images. This results in new rotation matrices, \( \hat{R} \), and intrinsic matrices, \( \hat{K} \), of the cameras. Because the original and new cameras are at the same position, images taken from them can be transformed one another through the Homography warping, as also illustrated in Fig. 6. Therefore, we compute the corresponding Homography transformation matrix and warp the images accordingly:
\[
\mathcal{W}(\hat{R}, \hat{K}) : \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \mapsto w\hat{K}\hat{R}^T RK^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}.
\]

Random cutouts are also applied to address occlusion.

For grasping and hanging, i.e., \( \text{task} \in \{\text{grasp}, \text{hang}\} \), we generate random poses \( q_{\text{task}} \in SE(3) \) in each iteration as a weighted sum of a (randomly picked) feasible pose and a random pose \( q_{\text{rand}} = t q_{\text{feasible}} + (1 - t) q_{\text{rand}}, t \sim \mathcal{U}(0, 1) \) where the position of \( q_{\text{rand}} \) is from the normal distribution and its quaternion is sampled uniformly, to encourage more precise prediction around the constraint manifolds. The training target is then, similarly to [5], the unsigned distances (in SE(3)) of \( q_{\text{task}} \) from the set of the feasible poses:
\[
d_{\text{task}} = \min_{j \in \{1, \ldots, 1000\}} ||q - q^j_{\text{task}}||_2.
\]

C. Loss Function

The whole architecture, backbone and three task heads, is trained end-to-end. In each iteration, we choose a minibatch of mugs for which a subset of augmented images with their camera parameters, \( \mathcal{V} = \{(\hat{T}, \hat{T}, \hat{K}), \ldots, (\hat{T}^{N_{\text{view}}}, \hat{T}^{N_{\text{view}}}, \hat{K}^{N_{\text{view}}})\} \), a subset of SDF data, \( \{(p^{1:N_{\text{SDF}}}, SDF^{1:N_{\text{SDF}}})\} \), and the grasping/hanging data, \( \{(q_{\text{grasp}}^{1:N_{\text{task}}}, q_{\text{hang}}^{1:N_{\text{task}}})\} \), are sampled. The images are encoded only once per iteration and then the SDF, grasping, hanging features are queried at the sampled points and poses. The overall loss is given as \( \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sdf}} + \mathcal{L}_{\text{grasp}} + \mathcal{L}_{\text{hang}} \), where we used a typical L1 loss for SDFs, i.e., \( \mathcal{L}_{\text{sdf}} = \frac{1}{N_{\text{SDF}}} \sum_{i=1}^{N_{\text{SDF}}} |\phi_{\text{sdf}}(p^i) - SDF^i| \), and the sign-agnostic L1 loss in [5] for grasping and hanging, i.e., \( \mathcal{L}_{\text{task}} \in \{\text{grasp}, \text{hang}\} \)
\[
\mathcal{L}_{\text{task}} = \frac{1}{N_{\text{task}}} \sum_{i=1}^{N_{\text{task}}} |\phi_{\text{task}}(q^i_{\text{task}}; \mathcal{V}) - d^i_{\text{task}}|.
\]
We used \( N_{\text{views}} = 4, N_{\text{SDF}} = 300, N_{\text{grasp}} = 100, N_{\text{hang}} = 100 \) and the considered interaction points are shown in Fig. 4.

V. SEQUENTIAL MANIPULATION PLANNING WITH DVC

In order to compute a full trajectory of the robot and objects that it interacts with, the learned features can be integrated as differentiable constraints into any constraint-based trajectory optimization framework, for which we adopt Logic-Geometric Programming (LGP) [3]. As depicted in Fig. 5, DVCs should be wrapped with multi-view preprocessing and forward kinematics to evaluate the learned constraints from the scene images and the optimization variable, i.e., robot’s joint configuration and object’s transformation.

A. Multi-View Preprocessing

In typical manipulation scenes, cameras are equipped as their views cover a wide range of the environment, so we need to transform the entire scene images into object-centric ones before passing them to the network. As illustrated in Fig. 6, multi-view processing finds a bounding ball and warps the raw images via the Homography warping. Let \( M^n \in \{0, 1\}^{W \times H} \) be the
are related by a homography. is computed as rigid objects, LGP is a hybrid steps per phase. A discrete action \( a \in \mathbb{R}^2 \). \( a = 2 \arcsin(2) \) The last baseline uses SDFs as object representations. The intrinsics as if the bounding sphere has a radius of finally warp the raw images accordingly. Regarding the shape reconstruction, we obtain an interaction feature as a function of a robot joint task sequence of discrete actions \( a \in \mathbb{R}^2 \), change \( r \), i.e., whether each mug is grasped or hung on a particular hook. Given a symbol sequence \( \{a_k, \delta q_{k-1,0}\} \) with \( s_k \in \mathbb{S}_{goal} \) from a logic tree search, we define the geometric path problem as a 2nd order Markov optimization [19]:

\[
\min_{\delta q_{1:KT}} \sum_{k=1}^{KT} f(x_{t-2:t}) ,
\]

s.t. \( \forall H \in \mathbb{H}(s_k, \delta q_{k-1,0}) \), \( H((x_{t-2:t}, \delta q_{k-1,0})_{t,i}) = 0 \),

where the initial joint states \( x_{-1,0} \) and objects’ transformations \( \delta q_{-1,0} = 0 \) are given. Note that \( \delta q \) denotes rigid transformations applied to objects’ implicit representations, not their absolute poses. \( f \) is a path cost that penalizes squared accelerations of the robot joints, but it can be more general if necessary. \( \mathbb{H}(s_k, \delta q_{k-1,0}) \) is a set of constraints the symbolic state and action impose on the geometric path at each phase \( k(t) = \lceil t/T \rceil \); these constraints include physical consistency, collision avoidance, and the learned interaction constraints that ensure the success of the discrete action \( a_k \). Lastly, \( I_H(s_k, \delta q_{k-1,0}) \) decides the time slice and object index that are subject to the constraint \( H \). As all the cost and constraint terms are differentiable and their Jacobians/Hessians are sparse, we can solve this constrained optimization problem efficiently using the augmented Lagrangian method with the Gauss-Newton approximation [19].

VI. EXPERIMENTS

A. Performance of Learned Features

Baselines: The key techniques of the proposed framework are threefold: the pixel-aligned technique, the implicit object representation over 3D and the task-guided learning scheme. To examine the benefits from each component, three baselines are considered. i) Global image features: The first baseline still represents an object as a function but the image encoder outputs a global image feature rather than having the pixel-aligned feature locally extracted; we used the ResNet-34 architecture as the image encoder and fixed the other model specifications. ii) Vector object representations: The second baseline represents an object as a finite-dimensional vector instead of a function; the representation network first computes the image features from the images using ResNet-34 and the camera features from the camera parameters using a couple of FCLs. Two features are then passed to another couple of FCLs to produce the object representation vector. The task heads take as input the feature’s pose as well as the object representation vector. iii) SDF representations: The last baseline uses SDFs as object representations; the network architecture for the SDF feature remains the same, but the grasping and hanging heads take as input a set of the keypoint’s SDF values instead of the d-dimensional representation vectors. The SDF values are detached when passed to the grasping/hanging heads so the backbone is trained by the geometry (SDF) data only. Evaluation Metric: Regarding the shape reconstruction, we report the Volumetric IoU and the Chamfer distance. To measure these metrics, we randomly sampled 4 images from the dataset and reconstructed the meshes from the learned SDF feature.
TABLE I

|                  | IoU       | Chamfer-$L_1$ ($\times 10^{-3}$) | Grasp+c (%) | Hang+c (%) |
|------------------|-----------|----------------------------------|-------------|------------|
| PIFO             | 0.816 / 0.656 | 5.26 / 6.90                   | 88.1 / 82.5 | 94.0 / 78.9 |
| Global Image Feature | 0.697 / 0.581 | 7.42 / 9.49                  | 82.7 / 75.7 | 91.2 / 78.2 |
| Vector Object Representation | 0.036 / 0.014 | 38.6 / 39.7                  | 0.5 / 0.4  | 0.0 / 0.0  |
| SDF Object Representation | **0.845 / 0.667** | **4.90 / 6.83**             | 67.9 / 64.3 | 3.7 / 4.3  |
| GT Mesh + HE     | -         | -                               | 62.8 / 75.0 | 94.9 / 92.9 |
| Recon. + HE      | -         | -                               | 66.7 / 42.9 | 78.2 / 60.7 |

Fig. 7. Predicted SDFs of an unseen mug (having a complex shape handle) from (b) PIFO and (c) the global image feature model.

using the marching cube algorithm. The volumetric IoU is the ratio between the intersection and the union of the reconstructed and ground-truth meshes which is (approximately) computed on the $100^3$ grid points around the objects. To compute the Chamfer distance, we sampled 10,000 surface points from each mesh and averaged the forward and backward closest pair distances. To evaluate the learned task features, we solved the unconstrained optimization $\hat{q}^* = \arg \min_q ||\phi_{\text{task}}(q)||^2$, $\text{task} \in \{\text{grasp, hang}\}$ using the Gauss-Newton method. Starting from this solution, we then solved the second optimization problem by including the collision feature, $\hat{q}^* = \arg \min_q ||\phi_{\text{task}}(q)||^2 + w_{\text{coll}}||\phi_{\text{coll}}(q)||^2$. Because the local optimization method can be stuck at local optima, we ran the algorithm from 10 random initial guesses in parallel and picked the best one. The optimized pose is finally tested in simulation and the success rates (feasibility) are reported in Table I.

Result: Table I shows that the SDF representation has the best shape reconstruction performance; PIFO is slightly worse, followed by the other two baselines. On the other hand, the task performances of PIFO are significantly better than the others. The SDF representation is especially worse in the hanging task, which implies that SDFs along the line are not sufficient for its feature prediction and our task-guided representation simplifies the feature prediction. In addition, it can be observed from Fig. 7 that the pixel-aligned method was better able to capture fine-grained details than the global image feature which reconstructed the handle shape as being more “typical”.

Hand-Engineered Constraint Models: We also compared our model to hand-engineered constraint models, iv) GT Mesh + HE and v) Recon. + HE, each of which computes constraint values based on the ground-truth meshes and the meshes reconstructed by the above SDF representations. Notably, Figs. 8(a)–(b) show how vulnerable the hand-engineered constraints can be to the reconstruction error; i.e., the error is directly associated with the planning result. While the perception pipeline for this geometric representation is never encouraged to reconstruct the “graspable/hangable parts” more accurately, we can view our end-to-end representation learning via task supervision as a way to do so. Moreover, the hand-engineered feature sometimes produces a wrong grasping pose even for the ground truth mesh (e.g., Fig. 8(c)). One can argue that a better interaction feature could be hand-designed by investigating the physics and kinematic structures more deeply, but that would require a huge amount of human insights/efforts and thus is inevitably less scalable. In contrast, our data-driven approach eliminates this procedure and directly learns the interaction constraint models from empirical success data of physical interactions.

B. Sequential Manipulation Planning Via LGP

We first considered a basic pick & hang task as shown in Fig. 9(a). The environment contains one robot arm, one hook, one mug and 4 cameras, and the interaction modes are constrained by the discrete action sequence of [(GRASP, gripper, mug), (HANG, hook, mug)]. 10 mugs were picked from each of the training and test data sets and their initial poses are randomized. Before solving the full trajectory optimization, we first optimized each feature as in Section VI-A and added small regularization terms using the optimized poses to guide the optimizer away from local optima.

Fig. 8. Some failure cases of hand-engineered features. (a) The hand-engineered feature lead the optimizer to hang the mug through the wrongly generated hole. (Green transparent meshes represent the ground truth.) (b) The handle disappeared in reconstruction, so this part would never be grasped and the mug never be hung. (c) The hand-engineered feature generated a wrong grasping pose on the ground truth mesh.

Fig. 9. Sequential manipulation scenarios.

2Before solving the full trajectory optimization, we first optimized each feature as in Section VI-A and added small regularization terms using the optimized poses to guide the optimizer away from local optima.
re-plan and execute when it failed, the success rates increased to 90% and 70%, respectively.

To showcase the long-horizon planning capability of LGP, we considered the following two scenarios: i) The three-mug scenario consists of 6 discrete phases with \([\text{GRASP, gripper, mug1}], (\text{GRASP, gripper, mug2}), (\text{GRASP, gripper, mug3})\). ii) The handover scenario has two arms at different heights and the target hook is placed high, requiring two arms to coordinate a handover motion with the discrete actions \([\text{GRASP, R_gripper, mug}], (\text{GRASP, L_gripper, mug}), (\text{GRASP, L_hook, mug3})\) \]. Figs. 9(b)–(c) show the last configurations of the optimized plans; we refer readers to the accompanying video for clearer views.

**Inverse Kinematics with Generative Models:** One important attribute of our framework is that, while most existing works train generative models that directly produce the interaction poses, ours models interactions as equality constraints where multiple constraints can be jointly optimized with other planning features. To see the benefits of such joint optimization, we considered the following inverse kinematics problems with a generative model: For the basic pick & hang and handover scenarios, we optimized each interaction pose separately as in Section VI-A and checked if these individually optimized poses are kinematically feasible when combined together, i.e., whether or not the inverse kinematics problems have a solution. Even though the mug’s initial pose was given such that the first gasping is ensured feasible, 53 out of 100 pairs of grasp and hang poses were infeasible for the pick & hang scenario and 86 out of 100 sets for the handover scenario, i.e., many of the individually sampled poses led to a collision or an infeasible robot configuration for hanging or handover. One failure case is depicted in Fig. 10. As the sequence length gets longer, not only should an exponentially larger number of planning problems be solved to find a set of feasible poses, but also the found poses are not guaranteed to be optimal. The joint optimization with our constraint models doesn’t raise such issues.

**C. Exploiting Learned Representations: 6D Pose Estimation and Zero-Shot Imitation**

Fig. 11 visualizes three components of the image feature vectors (the outputs of the U-net encoder) from the principal component analysis (PCA). It can be observed that each component represents a certain property of the objects, such as inside vs. outside, handle vs. other parts, or above vs. below. This enables the image-based pose estimation which we call feature-based closest point (FCP) matching, i.e., the problem of finding the relative pose of a target mesh w.r.t. a model mesh, without defining any canonical coordinate of the objects. Specifically, the FCP matching works as follows:

1. It first queries the backbone at \(10^3\) and \(5^3\) grid points around the target and the model, respectively.
2. For each model grid point, the target point is obtained such that their representations are closest.
3. Finally, it computes a \(SE(3)\) pose that minimizes the sum of the model-target pairwise Euclidean distances.

We compared this to the conventional iterative closest point (ICP) algorithm on point clouds, i.e., the problem of finding the relative pose minimizing the Euclidean distance of two sets of point clouds. The point clouds can be obtained from depth cameras (ICP) or on the surface of the meshes reconstructed via the learned SDF features (ICP2). The point clouds’ size was 1000. Fig. 12 shows the position and orientation errors when 131 mugs with random poses were tested. FCP performs much better especially in orientation because, notoriously, ICP easily gets stuck in local optima. A significant improvement was observed in F+ICP2 where we used the FCP results as starting points of ICP2; note that it can perform without depth images. Another important observation from Fig. 11 is that the semantics of representations are consistent across different objects as well, e.g. the handle parts of different mugs have similar representations. It implies that a pose of one object can be transferred into another through the representation. We therefore considered an image-based zero-shot imitation scenario, where the environment contains one robot arm, one target mug (filled with small balls) and 4 cameras as shown in Fig. 13. We manually designed a pouring motion for one mug and stored the images of pre- and post-pouring postures of the mug, \(\mathcal{V}_{\text{pre}} = (\mathcal{T}_{\text{pre}}, \mathbf{T}_{\text{pre}}, \mathbf{K}_{\text{pre}})\) and \(\mathcal{V}_{\text{post}} = (\mathcal{T}_{\text{post}}, \mathbf{T}_{\text{post}}, \mathbf{K}_{\text{post}})\), respectively. For a new mug, we solved LGP with \([\text{GRASP, gripper, mug}], (\text{PoseFCP, } \mathcal{V}_{\text{pre}}, \text{mug}), (\text{PoseFCP, } \mathcal{V}_{\text{post}}, \text{mug})\), where \(\text{PoseFCP, } \mathcal{V}_{\text{post}}\) imposes the aforementioned FCP constraint at the end of each phase.
That is, the trajectory optimizer tries to match each part of the new object to the corresponding part of the target mug while coordinating the global consistency of the full trajectory (e.g., determining a proper grasp pose for pouring). Fig. 13 shows the optimized post-pouring posture, which implies that the learned representation allows for imitation of the reference motions only from the posed images.

D. Real Robot Demonstration

Fig. 1 shows our complete framework in the real robot system. To successfully apply the learned DVCs to the real robot by closing the sim-to-real gap, we had to extend training to a larger dataset; specifically, we randomized the material of mugs by adjusting metalness and roughness to get more diverse appearances and also applied more extensive data augmentations, e.g., Color-Jitter or GaussianBlur. At test time, we attached RealSense D435 and also applied more extensive data augmentations, e.g. Color-Jitter or GaussianBlur. At test time, we attached RealSense D435

VII. DISCUSSION

The main idea of the proposed DVCs is twofold: i) Implicit object representations to which manipulation planning algorithms can apply rigid transformations in $SE(3)$ and ii) the implicit representations trained as a shared backbone of multiple task features, directly via task-supervisions. Throughout the experiments, we demonstrated the proposed visual manipulation framework both in simulation and with the real robot. The ablation studies examined each of the proposed techniques, compared to the non-pixel aligned, explicit, and geometric representations as well as the traditional hand-engineered features. The IK experiments demonstrated the advantage of DVC’s joint optimization capability. We analyzed the learned representations via PCA and found that the generalizable sementsics emerged in the representation during training, which enables 6D pose estimation and zero-shot imitation.

Notably, the last finding implies that considering more diverse tasks and objects in our multi-task learning would lead to more generalized representations as well as stronger synergies between individual feature learning, which we leave for future work. All those task features don’t necessarily model physical interaction feasibility for planning; e.g., they can also serve as a value or energy function of a direct control policy and be trained via imitation or reinforcement learning as well [21], [22].

REFERENCES

[1] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 165–174.
[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 405–421.
[3] M. Toussaint, K. Allen, K. A. Smith, and J. B. Tenenbaum, “Differentiable physics and stable modes for tool-use and manipulation planning,” in Proc. Robot.: Sci. Syst., 2018.
[4] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy networks: Learning 3D reconstruction in function space,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4460–4470.
[5] M. Atzmon and Y. Lipman, “SAL: Sign agnostic learning of shapes from raw data,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2565–2574.
[6] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, “Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 3504–3515.
[7] P. Florence, L. Manuelli, and R. Tedrake, “Dense object nets: Learning dense visual object descriptors by and for robotic manipulation,” in Proc. Conf. Robot. Learn., 2018, pp. 373–385.
[8] P. Florence, L. Manuelli, and R. Tedrake, “Self-supervised correspondence in visuomotor policy learning,” IEEE Robot. Automat. Lett., vol. 5, no. 2, pp. 492–499, Apr. 2020.
[9] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, “TossingBot: Learning to throw arbitrary objects with residual physics,” IEEE Trans. Robot., vol. 36, no. 4, pp. 1307–1319, Aug. 2020.
[10] A. Zeng et al., “Transporter networks: Rearranging the visual world for robotic manipulation,” in Proc. Conf. Robot. Learn., 2020, pp. 726–747.
[11] M. Breyer, J. J. Chung, L. Ott, S. Roland, and N. Juan, “Volumetric grasping network: Real-time 6 DoF grasp detection in clutter,” in Proc. Conf. Robot. Learn., 2021, pp. 1602–1611.
[12] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, “Synergies between affordance and geometry: 6-DoF grasp detection via implicit representations,” in Proc. Robot.: Sci. Syst., 2021.
[13] A. Simeonov et al., “Neural descriptor fields: SE(3)-equivariant object representations for manipulation,” in Proc. IEEE Int. Conf. Robot. Automat., 2022, pp. 6394–6400.
[14] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, “Learning models as functionals of signed-distance fields for manipulation planning,” in Proc. Ann. Conf. Robot. Learn., 2022, pp. 245–255.
[15] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint, “Learning multi-object dynamics with compositional neural radiance fields,” 2022, arXiv:2202.11855.
[16] A. X. Chang et al., “ShapeNet: An information-rich 3D model repository,” Stanford, CA, USA—Princeton University, Princeton, NJ, USA—Toyota Technological Institute at Chicago, Chicago, IL, USA, Tech. Rep., 2015.
[17] C. Eppner, A. Mousavian, and D. Fox, “ACRONYM: A large-scale grasp database based on simulation,” in Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 6222–6227.
[18] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” 2019, [Online]. Available: http://pybullet.org
[19] M. Toussaint, “A Tutorial on Newton Methods for Constrained Trajectory Optimization and Relations to SLAM, Gaussian Process Smoothing, Optimal Control, and Probabilistic Inference” in Geometric and Numerical Foundations of Movements. Berlin, Germany: Springer, 2017.
[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2961–2969.
[21] P. Florence et al., “Implicit behavioral cloning,” in Proc. Conf. Robot Learn., 2021, pp. 158–168.
[22] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint, “Learning geometric reasoning and control for long-horizon tasks from visual input,” in Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 14298–14305.