LEARNING NEURAL IMPLICIT FUNCTIONS AS OBJECT REPRESENTATIONS FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic manipulation planning is the problem of finding a sequence of robot configurations that involves interactions with objects in the scene, e.g., grasp, placement, tool-use, etc. To achieve such interactions, traditional approaches require hand-designed features and object representations, and it still remains an open question how to describe such interactions with arbitrary objects in a flexible and efficient way. Inspired by neural implicit representations in 3D modeling, e.g. NeRF, we propose a method to represent objects as neural implicit functions upon which we can define and jointly train interaction features. The proposed pixel-aligned representation is directly inferred from camera images with known camera geometry, naturally acting as a perception component in the whole manipulation pipeline, while at the same time enabling sequential robot manipulation planning.

1 INTRODUCTION

Intelligent agents should be able to interact with objects in the environment, such as grasping, placing an object or more general tool-use, to achieve a certain goal. In robotics, such instances are formalized as manipulation planning, a type of a motion planning problem that solves not only for the robot's own movement but also for the objects' motions subject to interaction constraints, and often tackled via hybrid optimization over symbolic states prescribing interaction modes and continuous geometric paths. To ground the interaction symbols, traditional approaches represent objects using meshes or combinations of shape primitives and describe interactions as hand-crafted constraints in terms of that representation. The approach with these traditional geometric representations has long-standing limitations in the variety of objects and interaction modes that can be handled for two reasons: (i) The complexity of such representations grows with the variations of object shapes, thereby making hand-engineering of the interaction features on top of those intractable. (ii) The representations have to be inferred from raw sensory inputs like images or point clouds, which inevitably induce estimation errors, and more accurate inference doesn't necessarily mean more precise feature prediction.

What is a good representation of an object? Considering the representation will be used to predict interaction features, we expect it to encode not just the object's geometric information but more task-specific ones, and 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 features, 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 interact with the mug through them in many different ways. Also from the perception aspect, good representations should be easy to infer from raw sensory inputs and should be able to trade their accuracy off in favor of the feature prediction.

To this end, this work takes a data-driven approach to learning the whole pipeline from camera images to interaction features end-to-end. The object representation acts as a bottleneck between the images to the interaction features and is shared across multiple feature predictions so it can have the aforementioned properties of good representations. Particularly, inspired by their recent great advances in 3D modeling, we transfer the idea of neural implicit representations to robotic manipulation applications. We represent an object as a neural implicit function over the 3D space upon which equality constraint features that return zero when feasible and non-zero otherwise are built to describe interactions between the robot and the object. At each time step of the planning horizon, the implicit function is queried at key interaction points which are rigidly attached to the robot frame. The resulting set of the representation vectors are then passed to the feature head network

to predict the interaction constraint value. Moreover, the proposed neural implicit function is pixel-aligned, meaning that the function is built from an arbitrary number of images with their camera poses and intrinsics such that the representation at a certain spatial location corresponds to the associated pixels of the images. These learned interaction features, together with other constraints about the robot's own motions, are used to formulate manipulation planning problems, allowing for computing trajectories that involve various types of interactions only from images. Furthermore, thanks to the representations' generalization, the learned features are directly applicable to manipulation tasks involving unseen objects. Demo videos: [playlist1], [playlist2], [playlist3].

2 Related Work

2.1 Neural Implicit Representations in 3D Modeling and View Synthesis

Neural implicit 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 (Mescheder et al., 2019; Songyou Peng, 2020) or signed distance fields (Park et al., 2019; Gropp et al., 2020; Atzmon & Lipman, 2020). 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.

There have been attempts to associate these 3D representations with 2D images using the principle of camera geometry. Exploiting the camera geometry in a forward direction, i.e., 2D projection of 3D representations, yields a differentiable image rendering procedure and this idea can be used to get rid of 3D supervisions. For example, Sitzmann et al. (2019); Niemeyer et al. (2020); Yariv et al. (2020) showed that the representation networks can be trained without the 3D supervision by defining a loss function to be difference between the rendered images and the ground-truth. Another notable application of this idea is view synthesis. Especially, neural radiance fields (NeRFs) proposed by Mildenhall et al. (2020) represent scenes as functions of 3D spatial location and 2D viewing direction to the view-dependent colors and densities, upon which they showed a differentiable rendering procedure can produce photorealistic images at any novel views. The NeRF approach has been extended by incorporating many other 3D priors/rendering techniques and now become one of the most powerful tools for realistic and fast rendering of virtual worlds (Garbin et al., 2021; Martin-Brualla et al., 2021; Barron et al., 2021; Lindell et al., 2021). Based on the differentiable rendering, Park et al. (2020); Chen et al. (2020); Yen-Chen et al. (2021) addressed unseen object pose estimation problems, where the goal is to find object's pose relative to the camera that produces a rendered image closest to the ground truth.

By conditioning 3D representations on 2D input images, one can expect the amortized encoder network to directly generalize to novel 3D geometries without requiring any test-time optimization. This can be done by introducing a bottleneck of a finite-dimensional *global* latent vector between the images and representations, but these global features often fail to capture fine-grained details of the 3D models (Songyou Peng, 2020). To address this, the camera geometry can be exploited in the inverse direction to obtain pixel-aligned *local* representations, i.e., 3D reprojection of 2D image features. Saito et al. (2019) and Xu et al. (2019) showed that the pixel-aligned methods can establish rich latent features because they can easily preserve high-frequency components in the input images. Furthermore, these methods are able to naturally fuse arbitrary multiple 2D views of the same model, allowing for feeding more input images to improve the output quality. Also, Yu et al. (2021) and Trevithick & Yang (2021) incorporated this idea within the NeRF framework and showed that their convolutional encoders have strong generalizations to models both in training and unseen categories.

While the above work investigates neural implicit functions to model shapes or appearances, we train them to model physical interaction feasibility and therby to provide a differentiable constraint model for robot manipulation planning. Our proposed framework also takes advantages of the pixel-aligned method which is especially beneficial in robotic manipulations where capturing fine-grained details is critical for many interactions such as establishing a contact or avoiding collisions with the object.

2.2 OBJECT/SCENE REPRESENTATIONS FOR ROBOTIC MANIPULATIONS

To eliminate the hand-engineering burden of representations/features for manipulation planning, several works have proposed data-driven approaches to learn object representations and/or interaction features which are conditioned on raw sensory inputs, like camera images or point clouds. Especially, such attempts have mostly been made for grasping of diverse objects. One popular approach is to train discriminative models for grasp assessments. For example, ten Pas et al. (2017) and Mahler et al. (2017) trained a neural network that, for given candidate grasp poses, predicts their grasp qualities from point clouds. In addition, Breyer et al. (2020) and Jiang et al. (2021) designed 3D convolutional networks that take as inputs a truncated SDF and candidate grasp poses and return the grasp affordances. Apart from the grasping, Zeng et al. (2020) addressed an object throwing scenario and proposed a framework where a convolutional neural network maps an input scene image and a target location to a grasping score map and a trowing velocity map. While the idea of predicting task affordances using neural networks is also shared by our work, one crucial difference is that most existing methods rely on either exhaustive search over the pose space or heuristic sampling schemes to find feasible poses, both of which posses scalability issues. In contrast, our framework takes advantage of a powerful optimization algorithm by enforcing the learned features to have a smooth optimization landscape, thus not suffering from the high dimensionality of the configuration space nor the combinatorial complexity which arises when sequencing multiple interaction modes.

On the other hand, neural networks also have been used as generative models. For example, Mousavian et al. (2019) and Murali et al. (2020) adopted the approach of conditional variational autoencoders (C-VAEs) to model the feasible grasp pose distribution conditioned on the point cloud. Sundermeyer et al. (2021) proposed a somewhat hybrid method, where the network densely generates grasp candidates within one forward path by assigning grasp scores and orientations to the point cloud. You et al. (2021) addressed the object hanging tasks from point clouds where the framework first makes dense predictions of the candidate poses among which one is picked and refined. Although these generative methods are capable of modeling complex multi-modal distributions and sampling multiple poses from those very rapidly, it is unclear how to incorporate them into general sequential manipulation planning that includes not only multiple interaction modes but also other robot motion constraints; e.g., choosing a sequence of the optimal poses from the generated ones causes another combinatorial complexity.

Another important line of research is learning and utilizing keypoint object representations. Manuelli et al. (2019); Gao & Tedrake (2021) represented objects using a set of 3D semantic keypoints and formulated manipulation problems in terms of such the keypoints. More recently, Qin et al. (2020); Turpin et al. (2021) extended the framework by proposing a self-supervised training scheme and by making the keypoints more task-specific. Similarly, Manuelli et al. (2020) described and learned the dynamic models of objects as a function of keypoints upon which a model predictive control scheme is used to generate robot motions. Despite their strong generalizations to unseen objects, the keypoint representations require corresponding manipulation problems, i.e. how the keypoints should be consumed in the problem, and/or semantics of the keypoints to be predefined.

The representation part of our framework is closely related to dense object descriptions proposed by Florence et al. (2018; 2019). The idea is to train fully-convolutional neural networks that maps a raw input image to pixelwise object representations and the authors have shown that the motion policies learned in terms of the dense descriptions directly generalize to unseen objects. Our proposed framework can be seen as an extension of this pixelwise representation to dense representations over the 3D space which can be seamlessly integrated into general sequential manipulation planning problems. Another difference is that the dense description is learned through the self-supervised correspondence training whereas ours is learned to be task-specific via the interaction data.

3 METHOD

3.1 OVERVIEW

As shown in Fig. 1, our proposed image-based feature prediction framework consists of two parts: the representation network, which we call a backbone, and the feature head networks. The backbone serves as an implicit functional representation of an object, which, conditioned on a set of posed images, outputs d-dimensional representation vectors at queried 3D spatial locations. The interaction feature predictions are made through the feature heads, where each feature head is fed on a set of

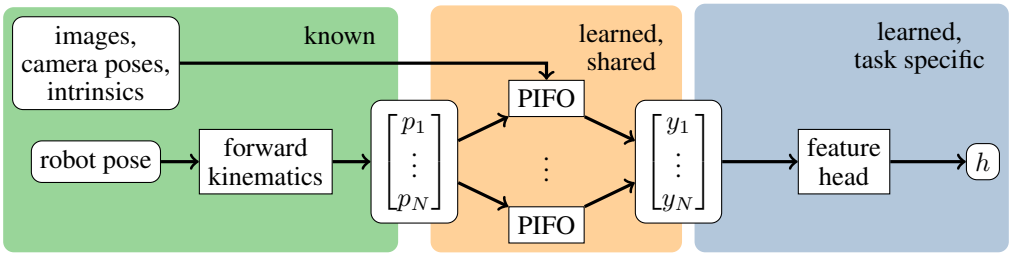


Figure 1: The proposed feature prediction scheme for manipulation planning

representation vectors obtained by querying the backbone at a set of representative points. Overall, the backbone is shared by the multiple feature heads and jointly trained with them. Sections 3.2 and 3.3 will be devoted to formally introduce each module. Section 3.4 presents how the learned feature representations are used to define a sequential manipulation planning problem and Section 3.5 discusses how the raw camera images can be transformed into focused ones.

3.2 PIXEL-ALIGNED IMPLICIT FUNCTIONAL OBJECT REPRESENTATION (PIFO)

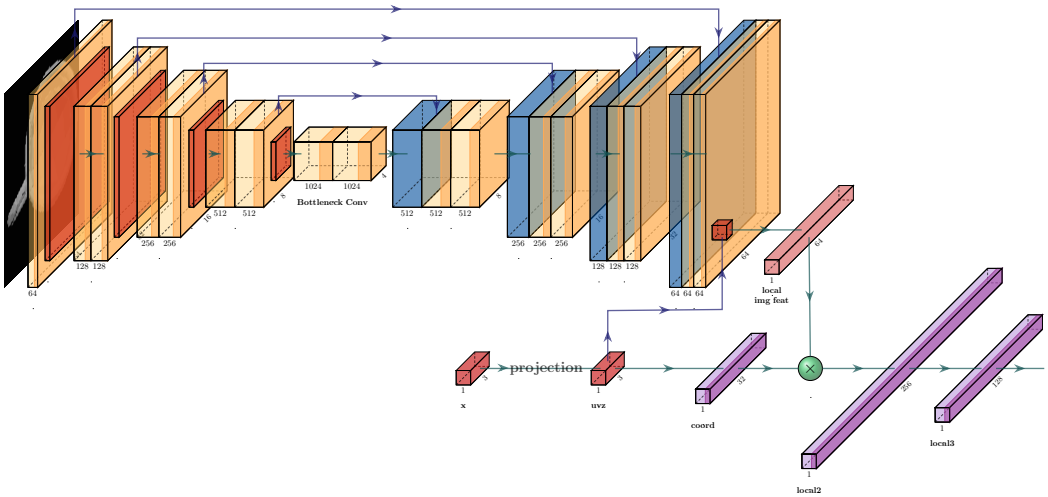


Figure 2: Image encoder and 3D reprojector of the backbone network.

Given a set of N images with their camera poses and intrinsics, $\{(\mathcal{I}^1, \boldsymbol{T}^1, \boldsymbol{K}^1), ..., (\mathcal{I}^{N_{\text{view}}}, \boldsymbol{T}^{N_{\text{view}}}, \boldsymbol{K}^{N_{\text{view}}})\}$, we define a functional representation of an object as a mapping:

$$\psi(\{(\mathcal{I}^1, \boldsymbol{T}^1, \boldsymbol{K}^1), ..., (\mathcal{I}^{N_{\text{view}}}, \boldsymbol{T}^{N_{\text{view}}}, \boldsymbol{K}^{N_{\text{view}}})\}, x) = y, \tag{1}$$

where $x \in \mathbb{R}^3$ is a queried 3D position and $y \in \mathbb{R}^d$ is a representation vector at that point. The representation network consists of three parts: image encoder, 3D reprojector, and feature aggregator. The first two modules compute a representation vector for each image, as depicted in Fig. 2, and the last one combines them into one vector. While the multiple feature heads separately model different interactions, the backbone network is shared across them, making the backbone learn more general object representations and in turn creating synergies in multiple feature learning.

Image Encoder: This module takes as input an image and computes a feature image. In order to capture both local and global information in the image, we adopted the U-net architecture (Ronneberger et al., 2015), especially with the pretrained ResNet34 (He et al., 2016) as its downward path and two residual 3×3 convolutions followed by up-convolution as the upward path, i.e.,

$$\mathcal{F}^n = UNet(\mathcal{I}^n), \ \forall n \in \{1, ..., N_{\text{view}}\}. \tag{2}$$

3D Reprojector: To endow the network with the multi-view consistency, all the 3D operations are performed in the view space. The 3D reprojector first converts the queried point, x, to the image coordinate including depth, $\pi(x; \boldsymbol{T}, \boldsymbol{K}) \in \mathbb{R}^3$ and then uses the projected point to extract the image

feature from the feature image via bilinear interpolation. Finally, the extracted image features and the coordinate features[1] are passed to a couple of fully connected layers to compute the object representation vectors, i.e.,

$$y^n = MLP(\mathcal{F}^n(z^n), z^n),\ z^n = \pi(x; \boldsymbol{T}^n, \boldsymbol{K}^n),\ \forall n \in \{1, ..., N_{\text{view}}\}, \tag{3}$$

where we used two hidden layers with the width of (256, 128).

Feature Aggregator: The feature aggregation from the multiple views should be permutation-invariant. While there are many options, like summation, averaging, or more sophisticated attention mechanisms, we simply take the averaging operation for it, i.e. $y = \frac{1}{N_{\text{view}}} \sum_n y^n$.

3.3 Feature Prediction

A feature head network predicts the interaction feature value via a set of representative points. To do so, we first attach a set of keypoints on the robot frame and query the representation network at those keypoint positions. The feature head network then takes as input the representation vectors and computes a predicted feature value through a couple of fully connected layers, i.e.,

$$h = MLP(y_1, ..., y_{N_{\text{keypoint}}}).$$

Note that when the considered feature is an SDF, only a single point needs to be queried for prediction, i.e., $N_{\text{keypoint}} = 1$, so the whole architecture reduces to the surface reconstruction module in Saito et al. (2019).

3.4 Manipulation Planning with Learned Features

In order to compute a full trajectory of the robot and the objects that it interacts with, the learned features can be integrated into a planning problem as differentiable interaction constraints. In particular, we adopt Logic-Geometric Programming (LGP) (Toussaint et al., 2018) as an optimization-based manipulation planning framework. For an n-joint robot and m rigid objects, LGP is a hybrid optimization problem over the number of phases K, a sequence of discrete actions $a_{1:K}$ and a geometric path $x_{1:KT} \in \mathcal{X}^{KT}$ where $\mathcal{X} \in \mathbb{R}^n \times SE(3)^m$ is the configuration space of the robot joint and the objects' pose. A discrete action a_k describes which interaction should be fulfilled at the end of the phase k, i.e., which mug to pick or on which hook to hang a grasped mug, and uniquely determines an interaction mode $s_k = \text{succ}(s_{k-1}, a_k)$, i.e., whether each mug is grasped or hung on a particular hook. Given $a_{1:K}$ and $s_{1:K}$ with $s_K \in \mathcal{S}_{\text{goal}}$, the lower level geometric path problem is defined as

$$\min_{x_{1:KT}} \sum_{t=1}^{KT} f_{\text{path}}\left(x_{t-2:t}, s_{k(t)}\right) \tag{4a}$$

$$\text{s.t. } \forall_{t \in \{1,...,KT\}} : h_{\text{path}}\left(x_{t-2:t}, s_{k(t)}\right) = 0,\ g_{\text{path}}\left(x_{t-2:t}, s_{k(t)}\right) \leq 0 \tag{4b}$$

$$\forall_{k \in \{1,...,K\}} : h_{\text{switch}}\left(x_{kt-2:kt}, a_k\right) = 0,\ g_{\text{switch}}\left(x_{kt-2:kt}, a_k\right) \leq 0, \tag{4c}$$

where f_{path} is a path cost penalizing squared accelerations of the robot joints, and $(h, g)_{\text{path}}$ define path constraints for a given mode s_k with $k(t) = \lfloor t/T \rfloor$ which ensure the optimized path to obey the appropriate physics and to be consistent with the mode, e.g., when grasped, a mug should be rigidly attached to the gripper. Most importantly, $(h, g)_{\text{switch}}$ define switch constraints for a given discrete action a_k and this is the place where the learned features are incorporated. Because all the cost/constraint terms are differentiable and their Jacobians/Hessians are sparse, we can solve this path optimization problem very efficiently using the Gauss-Newton optimization method.

3.5 Multi-View Preprocessing

In typical manipulation scenes, cameras are fixed and their views cover a wide range of the environment. Although the view space operations of the proposed framework and the data augmentation scheme enable the network to consider any relative poses of the cameras w.r.t. an object, feeding a raw image of the entire domain can cause huge computation burden and/or limits the input image resolution on the object. Similarly to the aforementioned image data augmentation, we warp the raw images to get the cropped images focusing on the object where the warping parameters are obtained from object masks.

[1] The coordinate feature is necessary to handle out-of-image query points.

Let $\mathcal{M}_n \in \{0,1\}^{W\times H}$ be the object masks available along with the raw images $\mathcal{I}_n$, $\forall n = 1, ..., N_{\text{cam}}$. We first solve the following optimization whose objective is to find a position and radius of the minimal bounding sphere such that the warped images contain all the object pixels in the original images:

$$\min_{\boldsymbol{p}\in\mathbb{R}^3, r\in\mathbb{R}^+} r, \quad \text{s.t. } \forall_{(u,v)\in\{(u',v',1);\mathcal{M}_n(u',v')=1,\forall n\in\{1,...,N_{\text{cam}}\}\}} : ||\mathcal{W}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{K}})(u,v)||_2 < 1, \tag{5}$$

where $\hat{\boldsymbol{R}}$ can be computed from $\boldsymbol{p}$ and the camera's position $\boldsymbol{t}$, and $\hat{\boldsymbol{K}}$ is obtained from $fov = 2\arcsin(||\boldsymbol{t}-\boldsymbol{p}||_2/r)$ from which the images can be warped:

$$\mathcal{W}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{K}}) : \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \mapsto w\hat{\boldsymbol{K}}\hat{\boldsymbol{R}}^T\boldsymbol{R}\boldsymbol{K}^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{6}$$

as shown in Appendix A.1. After solving the optimization, we fix the camera orientations $\hat{\boldsymbol{R}}$, change the intrinsics as if the bounding sphere has a radius of 15 cm (like the data augmentation) and finally warp the raw images accordingly. Fig. 8 shows the raw images from an example environment and the images warped by the multi-view processing.

4 TRAINING

In this paper, we consider table-top manipulation scenarios where a panda robot has to pick and hang mugs on hooks. The environment contains multiple mugs having different shapes and hooks on which a mug can be hung. To formulate and solve this manipulation problem, three types of interaction features are necessary: an SDF feature for collision avoidance, a grasping feature and a hanging feature. The grasping and hanging features are functions of the gripper's and the hook's relative poses, respectively, w.r.t. a mug.

4.1 DATA GENERATION

We took 131 mesh models of mugs from ShapeNet (Chang et al., 2015) 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 datasets of the posed images and each interaction as follows.

Posed Images: The posed image data consists of 100 images (128×128) with their corresponding camera poses and intrinsic matrices generated by an OpenGL renderer. Azimuths and elevations of the cameras are sampled such that they can uniformly be distributed on the surface of a sphere, while their distances from the object center are randomly chosen. The azimuth, elevation and distance fully determine the camera's positions and making them upright and face the object center gives the camera's orientations. For the intrinsics, we used $fov = 2\arcsin(d/r)$, where d is the camera distance and r is the half of the object size, so that the entire object appears in the image. Lighting is also randomized.

SDF: We sampled 12,500 positions from the 3D space and computed their signed distance values. Following the approach of DeepSDF (Park et al., 2019), we sampled more aggressively near the object surface to encourage the network to learn the object geometry more precisely.

Grasping & Hanging: We obtained 1,000 feasible grasping and hanging poses, respectively. For the grasping data, we used an antipodal sampling scheme, similarly to Eppner et al. (2021), to create candidate gripper poses and checked their feasibilities using the Bullet simulator (Coumans & Bai, 2016–2021). For the hanging data, we randomly sampled collision-free candidate hook poses and kinematically checked if the hook is trapped by the mug in directions perpendicular to the hook's main axis.

Figs. 6 (a)–(e) show some rough looks of the generated data. In the end, we have a dataset of:

$$\mathcal{D} = \left\{ \left(\mathcal{I}^{1:100}, \boldsymbol{T}^{1:100}, \boldsymbol{K}^{1:100}, x^{1:12500}, SDF^{1:12500}, q_{\text{grasp}}^{1:1000}, q_{\text{hang}}^{1:1000}\right)^{(i)} \right\}_{i=1}^{131}. \tag{7}$$

4.2 DATA AUGMENTATION AND LOSS FUNCTION

While randomizing the azimuth, elevation and distance of the camera provides all possible appearances of the object, it still cannot account for the roll angles of the camera 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 use a data augmentation technique based on Homography warping: In each iteration, for a randomly sampled set of images, we artificially perturb the estimated object center position (at which the cameras are looking) and the roll angle of each camera. Also, fov is modified as if the radius of the bounding sphere is 15 cm so that smaller objects can look smaller in the augmented images. This gives us new rotation matrices, $\hat{\boldsymbol{R}}$, and intrinsic matrices, $\hat{\boldsymbol{K}}$, of the cameras. We then compute the corresponding Homography transform matrices as shown in Appendix A.1 and warp the images accordingly. In addition, we apply a random cutout to address cases when the object is occluded. Figs. 6 (f) and (g) show how the image augmentation works.

For grasping and hanging feature learning, we generate 6D random poses $q_{\text{rand}} \sim \mathcal{P}$ in each iteration[2] and, similarly to Atzmon & Lipman (2020), set the training target to be unsinged distances to the set of the feasible poses:

$$d(q; q_{\text{task}}^{1:N_{\text{task}}}) = \min_{i=1,\dots N_{\text{task}}} ||q - q_{\text{task}}^{i}||_2. \tag{8}$$

For the network training, we first choose a minibatch of mugs from which a subset of augmented images with their camera poses and intrinsics, $\left(\hat{\mathcal{I}}^{1:N_{\text{views}}}, \hat{\boldsymbol{T}}^{1:N_{\text{views}}}, \hat{\boldsymbol{K}}^{1:N_{\text{views}}}\right)$, a subset of SDF data, $\left(x^{1:N_{\text{SDF}}}, SDF^{1:N_{\text{SDF}}}\right)$, and the grasping/hanging data, $\left(q_{\text{rand}}^{1:N_{\text{task}}}, d_{\text{task}}^{1:N_{\text{task}}}\right)$, are sampled. The images are encoded only once per iteration and then the SDF, grasping, hanging features are queried at the sample 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 a typical $L1$ loss is used for SDFs, i.e. $\mathcal{L}_{\text{sdf}} = \frac{1}{N_{\text{SDF}}} \sum_{i=1}^{N_{\text{SDF}}} |\phi_{\text{sdf}}(x^i) - SDF^i|$, and a loss of the sign-agnostic learning (Atzmon & Lipman, 2020) for grasping and hanging, i.e.,

$$\mathcal{L}_{\text{task}} = \frac{1}{N_{\text{task}}} \sum_{i=1}^{N_{\text{task}}} \left| \left|\phi_{\text{task}}(q_{\text{rand}}^{i})\right| - d_{\text{task}}^{i} \right|, \ \forall \text{task} \in \{\text{grasp, hang}\}. \tag{9}$$

The feature head and backbone networks are trained end-to-end. For training, we used $N_{\text{views}} = 4,\ N_{\text{SDF}} = 300,\ N_{\text{grasp}} = 1000,\ N_{\text{hang}} = 1000$. As shown in Fig. 7, the grasp and hang interaction keypoints are defined as $(3 \times 3 \times 3)$ grid points around the gripper center and as 5 points along the hook's main axis, respectively.

5 EXPERIMENTS

5.1 PERFORMANCE OF LEARNED FEATURES

The whole dataset for the 131 meshes is divided into 78 training, 25 validation and 28 test sets. The feature dimension is 128. The SDF feature head has one hidden layer whose width is 128. The grasp and hang heads have two hidden layers with the widths of $(256, 128)$. The network was trained until the validation loss stopped decreasing.

Regarding the shape reconstruction, Table 1 shows the Volumetric IoU and the Chamfer distance. To measure these metrics, we first randomly sampled 4 images from the dataset and reconstructed the meshes from the learned SDF feature using the marching cube algorithm. The volumetric IoU is the ratio between the interaction and the union of the reconstructed and ground-truth meshes which is (approximately) computed on the grid points around the objects. To compute the Chamfer distance, we sampled 10,000 points from the surfaces of each mesh and averaged the forward and backward closest pair distances. The SDF errors in Table 2 were also measured at the grid points.

To evaluate the learned task features, we solved the unconstrained optimization problems of $\min_q ||\phi_{\text{task}}(q)||^2,\ \text{task} \in \{\text{grasp, hang}\}$ using the 2nd-order Gauss-Newton method. Starting from these solutions, we then solved the second optimization problems by including the collision feature, which can be defined using the learned SDF feature, $\min_q ||\phi_{\text{task}}(q)||^2 + w_{\text{coll}}||\phi_{\text{coll}}(q)||^2$. 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 resulting feasible rates are shown in Table 1.

[2]To encourage more precise prediction around the constraint manifolds, we used a weighted sum of a feasible pose and a random pose $q_{\text{rand}} = tq_{\text{feasible}} + (1-t)q_{\text{normal}},\ t \sim \mathcal{U}(0,1)$ where the position of q_{normal} is from the normal distribution and its quaternion is sampled uniformly.

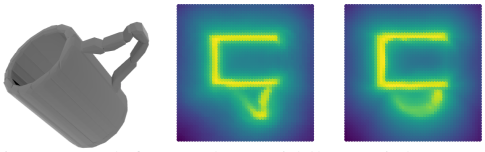


Figure 3: (left) Mesh. (middle & right) SDFs predicted by PIFO and the global latent model.

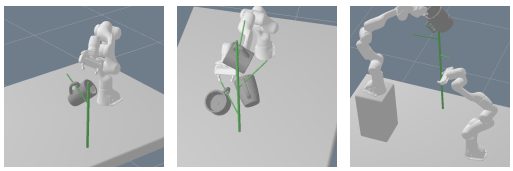


Figure 4: Sequential Manipulation scenarios: Single-, three-mug hanging and handover.

	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 Latent	0.697 / 0.581	7.42 / 9.49	82.7 / 75.7	91.2 / 78.2
Vector Repr.	0.036 / 0.014	38.6 / 39.7	0.5 / 0.4	0.0 / 0.0
SDF Repr.	**0.845 / 0.667**	**4.90 / 6.83**	67.9 / 64.3	3.7 / 4.3
PIFO (2 views)	0.760 / 0.577	6.14 / 8.84	82.9 / 77.1	88.2 / 72.1
PIFO (8 views)	0.851 / 0.683	4.78 / 6.34	88.7 / 85.0	96.5 / 82.5

Table 1: Individual Feature Evaluation with 4 images (Training / Test).

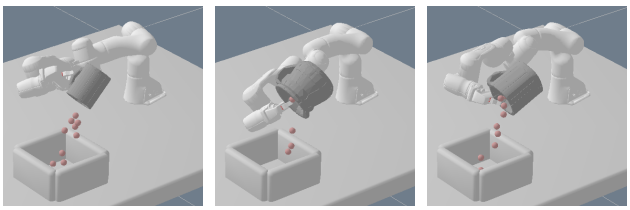
5.1.1 COMPARISONS

The key techniques of the proposed framework, *PIFO*, are threefold: the pixel-aligned latent inputs from the image encoder, the implicit function over the 3D space as representations and the task guided learning scheme. To examine the effectiveness of each component, three baselines are considered: (i) global latent vectors and (ii) vector object representations and (iii) SDF representations. The first baseline is the ablation of the pixel-aligned technique where the object is still represented as an implicit function but the image encode outputs a global latent vector; we used the ResNet34 architecture as the image encoder and fixed the other model specifications. The second baseline represents the object as a finite-dimensional vector instead of an implicit function; the representation network first computes the image features from the images using ResNet34 and the camera features are computed from the camera poses and intrinsics using a couple of fully connected layers. Two features are then passed to another couple of fully connected layers to produce the object representation vector. 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 inputs a set of the keypoints' SDF values instead of the d-dimensional representation vectors. Also, the SDF values are detached when passed to the heads so the backbone is only trained by the geometry (SDF) data.

Table 1 shows that the SDF representations have the best shape reconstruction performance; PIFO is slightly worse, followed by the other two frameworks. On the other hand, the task performances of PIFO are significantly better than the others. The SDF representations are especially worse in the hanging task, which implies that SDFs along the line are not sufficient for the feature prediction while our task-guided representations simplify the feature prediction. Fig. 3 depicts SDF values of an unseen mug with a complex shape handle predicted by PIFO and the global latent model; one can observe that the pixel-aligned representations are better able to capture fine-grained details of unseen objects. The networks were also tested with the different numbers of input images and it can be seen from the last two rows of the table that the more images we put in, the better performance the network shows. Tables 2 and 3 report all combinations of the metrics and the number of views.

5.2 SEQUENTIAL MANIPULATION PLANNING VIA LGP

The learned features can be used to formulate and solve sequential manipulation planning. Here, we considered three manipulation scenarios shown in Fig. 4: Basic pick & hang, long-horizon three-mug hanging, and handover. In the first scenario, the environment contains one robot arm (franka panda), 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)] within the LGP framework (Toussaint et al., 2018). 10 mugs were picked from each of the training and test data sets and their initial poses are randomized. (GRASP) induces the equality constraints on the gripper's approaching direction (for collision-free grasping), zero-impact switching and the learned grasping constraint. Similarly, (HANG) consists of the hanging approaching constraints, zero-impact and the learned hanging constraint. The collision constraints are also imposed, where the pair collisions with the mug are computed by the learned SDF feature. Before solving the full trajectory optimization, we first optimized each feature as in Sec. 5.1 and added small regularization terms using the optimized poses to guide the optimizer away from local optima. When executed the optimized trajectory in the



(a) (b) (c)

Figure 5: Zero-shot Imitation. (a) Reference Motion (b-e) imitation

Bullet simulation, the success rates on the train and test mugs were 50 % and 40 %, respectively. If we allow the method to re-plan and execute when it failed, the success rates increased to 90% and 70%, respectively [playlist1].

The three-mug scenario consists of 6 discrete phases with the skeleton of [(GRASP, gripper, mug1), (HANG, M_hook, mug1), (GRASP, gripper, mug2), (HANG, U_hook, mug2), (GRASP, gripper, mug3), (HANG, L_hook, mug3)]. The handover scenario has two arms at different heights and the target hook is placed very high, requiring two arms to coordinate a handover motion. The corresponding skeleton is given as [(GRASP, R_gripper, mug), (GRASP, L_gripper, mug), (HANG, U_hook, mug)]. Fig. 4 shows the last configurations of the optimized trajectories; we refer the reviewer to Figs. 11–12 and the accompanying videos [playlist2] for clearer views of the optimized motions. The learned features allowed for formulating and solving the challenging manipulation planning problems only from images.

5.3 ZERO-SHOT IMITATION

Figs. 9 and 10 visualize the first 5 principal components of the image features and the first principal components of the representation vectors, respectively. It can be seen that each component represents a certain property of the objects, such as inside vs. outside, handle vs. other parts or above vs. below, and its semantic is consistent across different objects. Similar to the idea of category-level pose estimation, a pose of one object can be transferred into a pose of another through this representation. We considered the image-based zero-shot imitation scenario, where the environment contains a robot arm, one target mug (filled with small balls) and 4 cameras as shown in Fig. 5. We manually designed the pouring motion and captured the camera images of pre- and post-pouring postures of the mug, $\mathcal{S}_{\text{pre}} = (\mathcal{I}_{\text{pre}}, \boldsymbol{T}_{\text{pre}}, \boldsymbol{K}_{\text{pre}})$ and $\mathcal{S}_{\text{post}} = (\mathcal{I}_{\text{post}}, \boldsymbol{T}_{\text{post}}, \boldsymbol{K}_{\text{post}})$, respectively. For new environments with a model mug, we solved the sequential manipulation planning with the skeleton [(GRASP, gripper, mug), (POSEICP, $\mathcal{S}_{\text{pre}}$, mug), (POSEICP, $\mathcal{S}_{\text{post}}$, mug)].

The discrete action POSEICP imposes the constraints of the iterative closest algorithm as follows: Grid points are attached both at the centers of the target and the actual bounding balls and are queried to the backbone with their own images. For each model points, the corresponding target point is obtained according to the distance in the feature space and their Euclidean distance is the feature value of this constraint; i.e., the optimizer tries to match each part of the model object to the corresponding part of the target while coordinating the global consistency over the full trajectory, resulting in the imitation of the reference motions *only* from the posed images. Fig. 5 shows the post-pouring posture of the target and imitated mugs; Figs. 13 – 14 and the accompanying videos [playlist3] show the reference and its imitations more clearly.

6 CONCLUSION

This work transferred the idea of neural implicit representations to robotic manipulation applications. The pixel-aligned nature enables the proposed representations to be inferred from posed camera images and easily capture fine-grained details of the objects. We demonstrated that the learned representations and interaction features allowed for formulating and solving general sequential manipulation problems only from images within the LGP framework. The learned representations generalize to unseen objects, enabling manipulation planning and zero-shot imitations for them.

REFERENCES

Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2565–2574, 2020.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Michel Breyer, Jen Jen Chung, Lionel Ott, Siegwart Roland, and Nieto Juan. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In *Conference on Robot Learning*, 2020.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *European Conference on Computer Vision*, pp. 139–156. Springer, 2020.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *2021 IEEE Int. Conf. on Robotics and Automation, ICRA*, 2021.

Peter Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *Conference on Robot Learning*, 2018.

Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.

Wei Gao and Russ Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 6(2):2962–2969, 2021.

Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pp. 3789–3799. PMLR, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, Yuke Zhu, Jonathan Tremblay, Stan Birchfield, Guanya Shi, Fabio Ramos, Animashree Anandkumar, et al. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Adam R Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Soňa Mokrá, and Danilo J Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *International Conference on Machine Learning*, 2021.

David B. Lindell, Julien N.P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 2017.

Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *International Symposium on Robotics Research (ISRR)*, 2019.

Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *Conference on Robot Learning*, 2020.

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4460–4470, 2019.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pp. 405–421. Springer, 2020.

Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2901–2910, 2019.

Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6232–6238. IEEE, 2020.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3504–3515, 2020.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019.

Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10710–10719, 2020.

Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7278–7285. IEEE, 2020.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2304–2314, 2019.

Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32:1121–1132, 2019.

Lars Mescheder Marc Pollefeys Andreas Geiger Songyou Peng, Michael Niemeyer. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

Marc Toussaint, Kelsey Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems*, 2018.

Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Dylan Turpin, Liquan Wang, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Gift: Generalizable interaction-aware functional tool affordances without labels. In *Robotics: Science and Systems*, 2021.

Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in Neural Information Processing Systems*, 32:492–502, 2019.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

Yifan You, Lin Shao, Toki Migimatsu, and Jeannette Bohg. Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319, 2020.

A Appendix

A.1 Homography Transformation

The idea of the Homography warping is that two images taken by cameras at the same position but with different orientations and intrinsics can be transformed into each other. Suppose that we have a source image $\mathcal{I}$ with the camera position $\boldsymbol{t}$, rotation matrix $\boldsymbol{R}$ and projection matrix $\boldsymbol{K}$ and that an object is inside a bounding sphere at $\boldsymbol{p} \in \mathbb{R}^3$ with a radius $r \in \mathbb{R}^+$. An image focusing on the bounding box can be taken from a (synthetic) camera at the same position $\boldsymbol{t}$ with the view direction as $\boldsymbol{t} - \boldsymbol{p}$ and the field of view angle as $2\arcsin(||\boldsymbol{t} - \boldsymbol{p}||_2/r)$, from which we can compute the new camera rotation matrix $\hat{\boldsymbol{R}}$ and the intrinsic $\hat{\boldsymbol{K}}$.

Given $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{K}}$, the new field warped by the corresponding Homography can be obtained as follows: First, a pixel in the source image, $p_1 = (u, v, 1)$, is reprojected into a ray in the 3D space: $P_1 = \boldsymbol{K}^{-1}p_1$. Next, the ray is viewed in the new camera coordinate: $P_2 = \hat{\boldsymbol{R}}^T\boldsymbol{R}P_1$. Lastly, this ray is projected back into a pixel in the new camera: $p_2 = \hat{\mathbf{K}}P_2$. Putting all together, the Homography warping is given as:

$$\mathcal{W}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{K}}) : \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \mapsto w\hat{\boldsymbol{K}}\hat{\boldsymbol{R}}^T\boldsymbol{R}\boldsymbol{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{10}$$

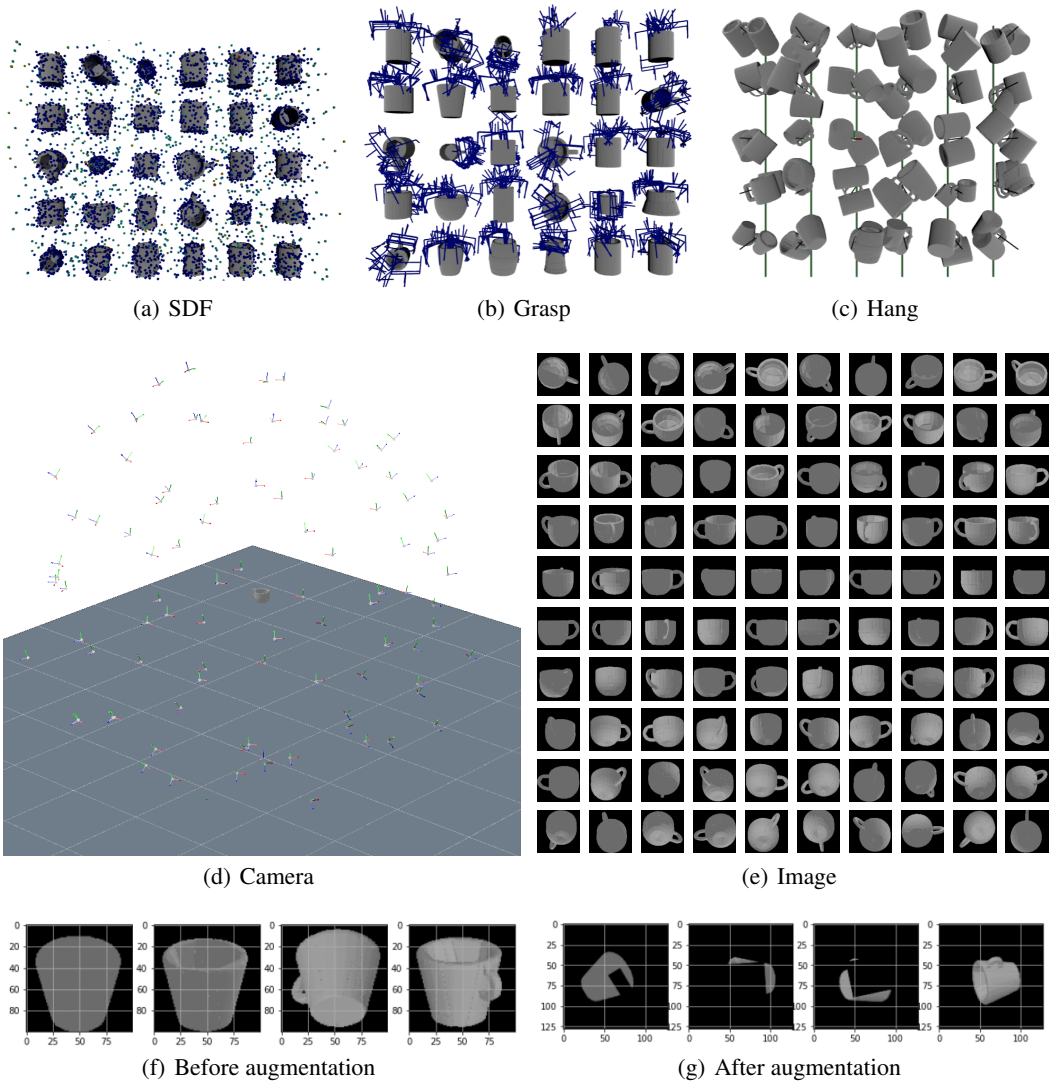
resulting in the warped image $\hat{\mathcal{I}}$ with its camera pose $\hat{\boldsymbol{T}} = \begin{bmatrix} \hat{\boldsymbol{R}} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}$ and intrinsic $\hat{\boldsymbol{K}}$.

# of views	Method	SDF error ($\times 10^{-3}$)	Volumetric IoU	Chamfer-L_1 ($\times 10^{-3}$)
2	PIFO	2.91 / 4.63	0.760 / 0.577	6.14 / 8.84
	Vector Latent	3.58 / 4.96	0.642 / 0.515	8.50 / 10.8
	Vector Representation	15.6 / 15.8	0.045 / 0.046	39.1 / 40.4
	SDF Representation	**2.11 / 3.48**	**0.786 / 0.622**	**5.78 / 8.13**
4	PIFO	2.20 / 3.38	0.816 / 0.656	5.26 / 6.90
	Vector Latent	2.82 / 3.93	0.697 / 0.581	7.42 / 9.49
	Vector Representation	15.0 / 15.2	0.036 / 0.014	38.6 / 39.7
	SDF Representation	**1.43 / 2.73**	**0.845 / 0.667**	**4.90 / 6.83**
8	PIFO	1.68 / 2.72	0.851 / 0.683	4.78 / 6.34
	Vector Latent	2.31 / 3.51	0.728 / 0.607	6.75 / 8.80
	Vector Representation	14.6 / 15.3	0.033 / 0.006	38.7 / 40.6
	SDF Representation	**1.07 / 2.07**	**0.878 / 0.703**	**4.51 / 6.06**

Table 2: SDF Feature Evaluation (Training / Test).

# of views	Method	Grasp (%)	Grasp+c (%)	Hang (%)	Hang+c (%)
2	PIFO	65.8 / 55.4	**82.9 / 77.1**	87.2 / **71.4**	**88.2 / 72.1**
	Vector Latent	**67.6 / 63.9**	80.9 / 70.4	**88.3** / 70.4	86.3 / 71.8
	Vector Representation	13.2 / 12.9	0.8 / 0.4	25.6 / 21.8	0.0 / 0.0
	SDF Representation	41.2 / 55.3	49.6/ 45.7	2.6 / 1.1	3.3 / 2.1
4	PIFO	**69.0 / 63.9**	**88.1 / 82.5**	88.7 / 75.4	**94.0 / 78.9**
	Vector Latent	62.3 / 61.8	82.7 / 75.7	**90.3 / 75.7**	91.2 / 78.2
	Vector Representation	21.2 / 22.5	0.5 / 0.4	55.1 / 46.4	0.0 / 0.0
	SDF Representation	49.1 / 46.1	67.9 / 64.3	3.3 / 2.9	3.7 / 4.3
8	PIFO	**71.9 / 69.3**	**88.7 / 85.0**	**91.7 / 80.4**	**96.5 / 82.5**
	Vector Latent	71.3 / 67.1	84.0 / 79.3	91.3 / 77.5	92.9 / 80.4
	Vector Representation	29.0 / 23.9	0.5 / 0.7	65.9 / 49.6	0.0 / 0.0
	SDF Representation	51.4 / 52.1	75.5 / 70.4	4.6 / 6.1	6.3/ 5.7

Table 3: Task Feature Evaluation (Training / Test).



(a) SDF (b) Grasp (c) Hang

(d) Camera (e) Image

(f) Before augmentation (g) After augmentation

Figure 6: Data Generation and Augmentation

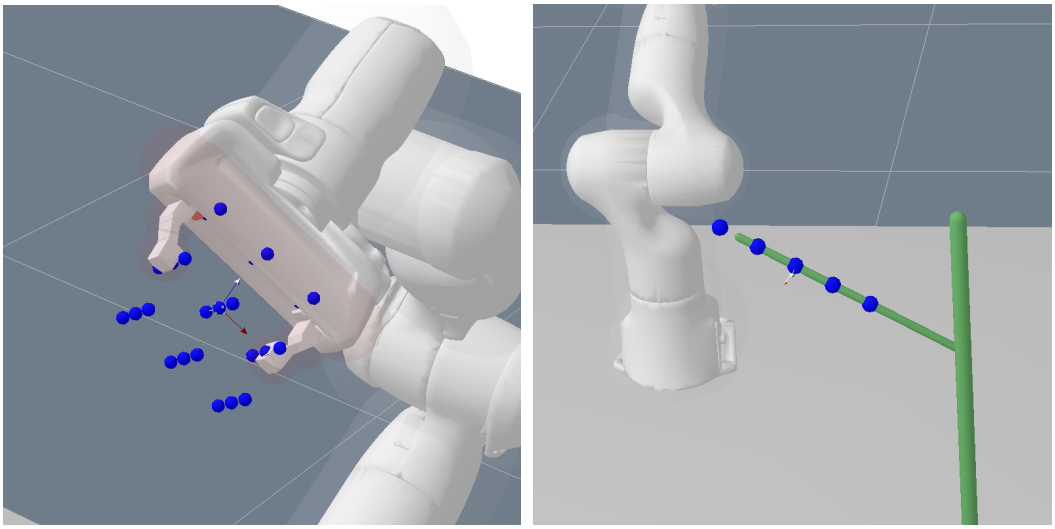
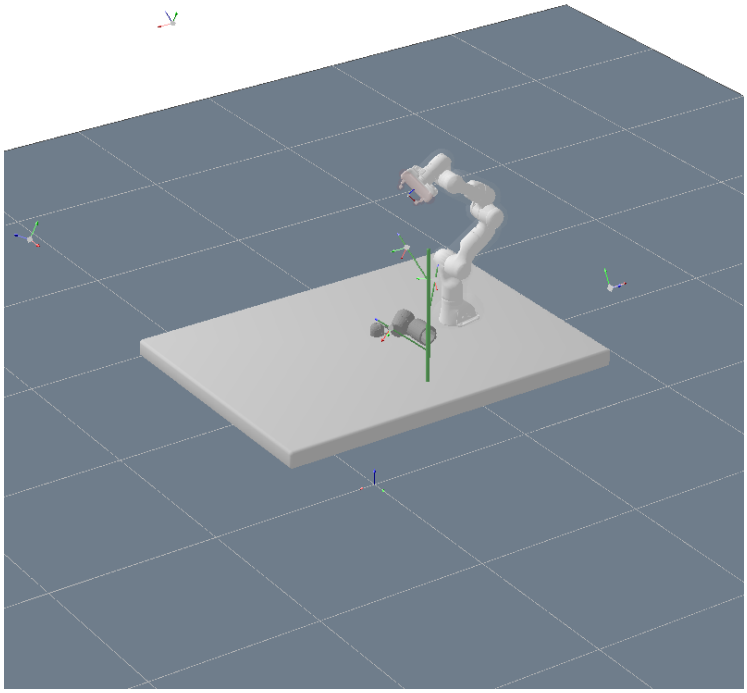
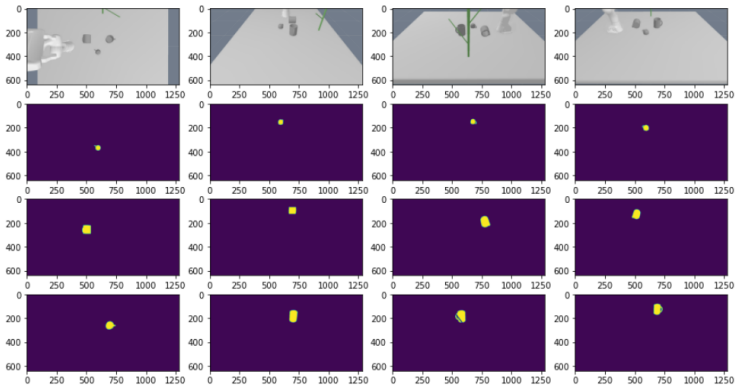


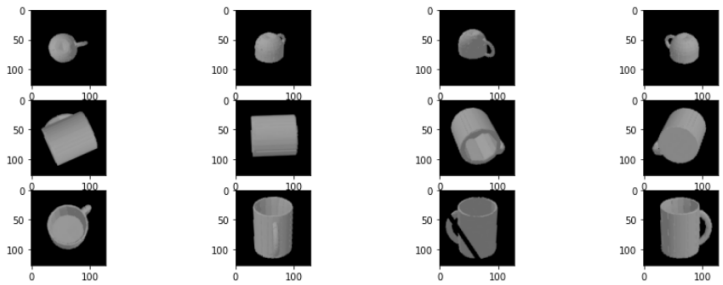
Figure 7: Key interaction points on the gripper and hook



(a) Scene

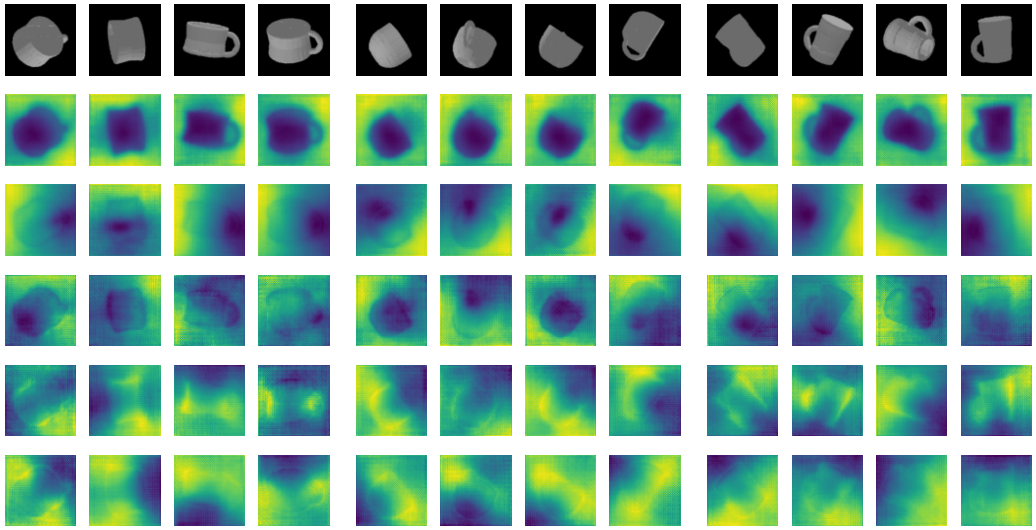


(b) Images and masks



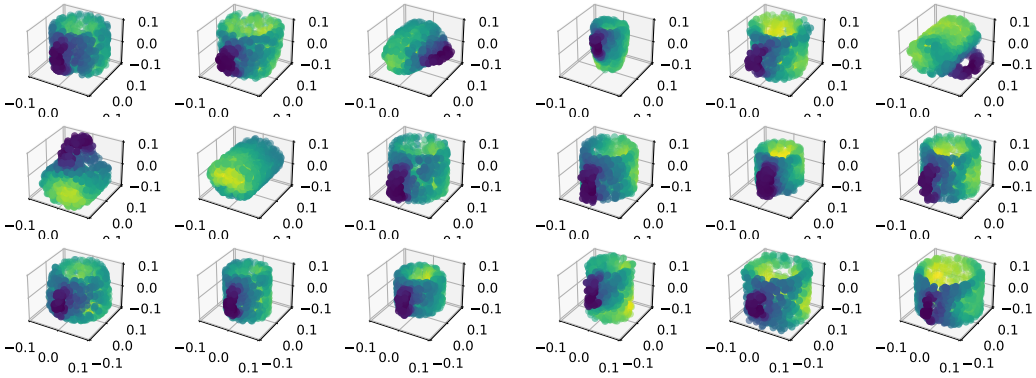
(c) Warped images (via the multi-view processing)

Figure 8: Multi-view processing



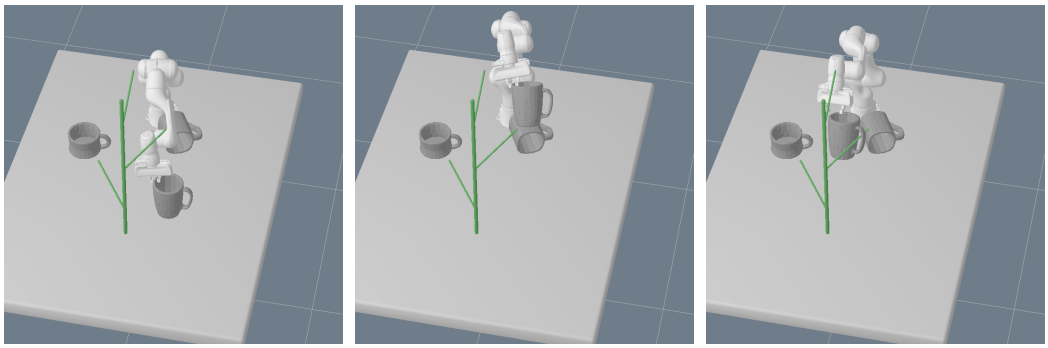
(a) (b) (c)

Figure 9: First 5 principal components from PCA on image features



(a) Train mugs (b) Test mugs

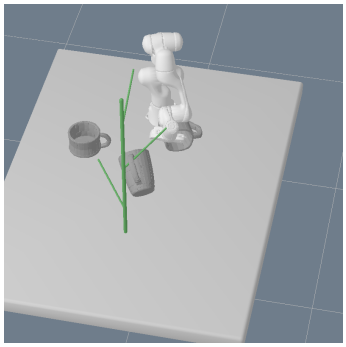
Figure 10: The first principal component from PCA on representation vectors of the 3D surface points



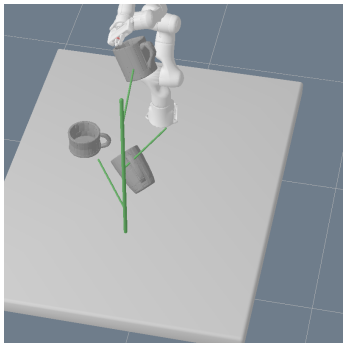
(a) t=5

(b) t=8.5

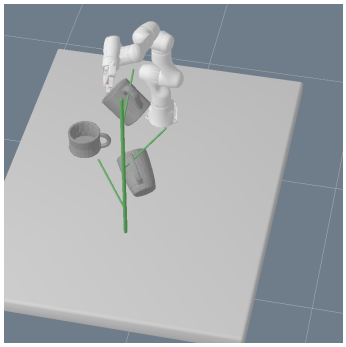
(c) t=10



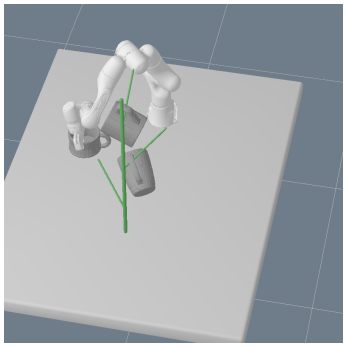
(d) t=15



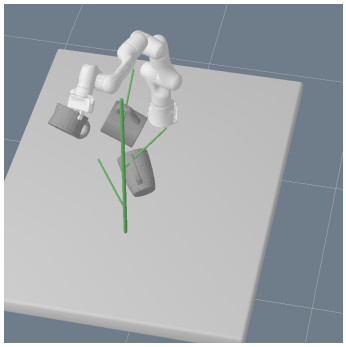
(e) t=18.5



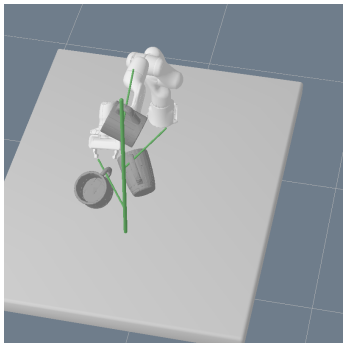
(f) t=20



(g) t=25

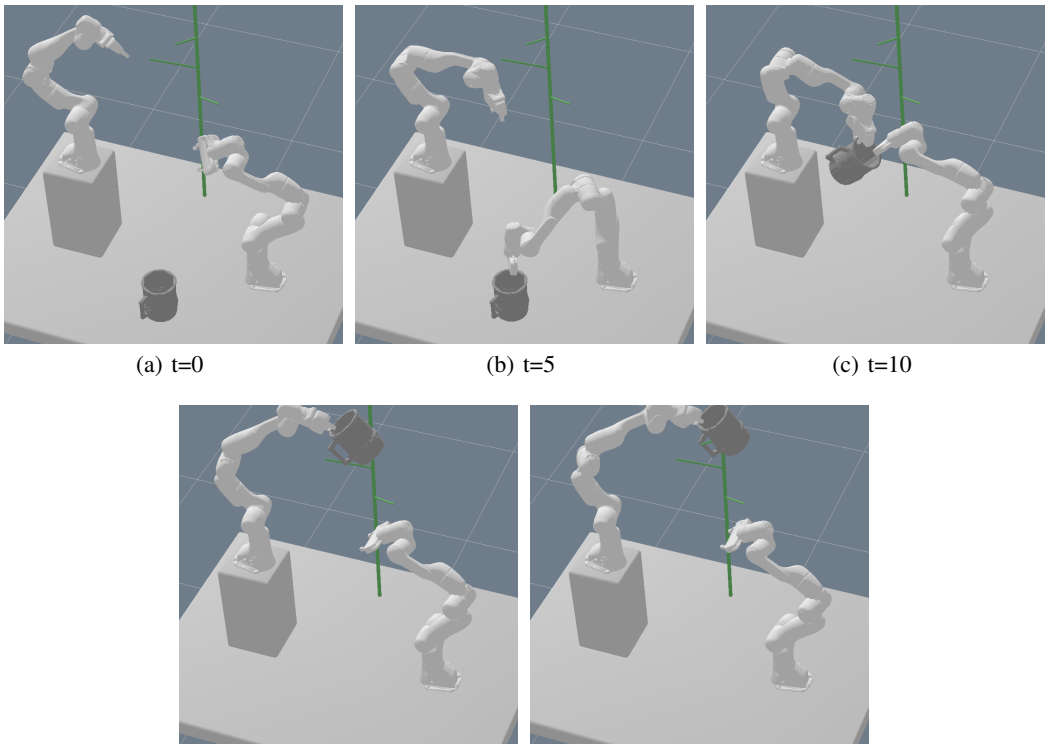


(h) t=28.5



(i) t=30

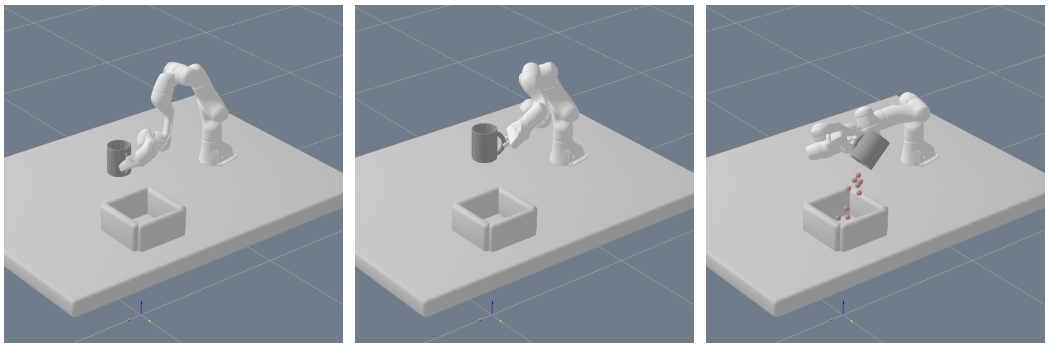
Figure 11: The three-mug scenario



(a) t=0 (b) t=5 (c) t=10

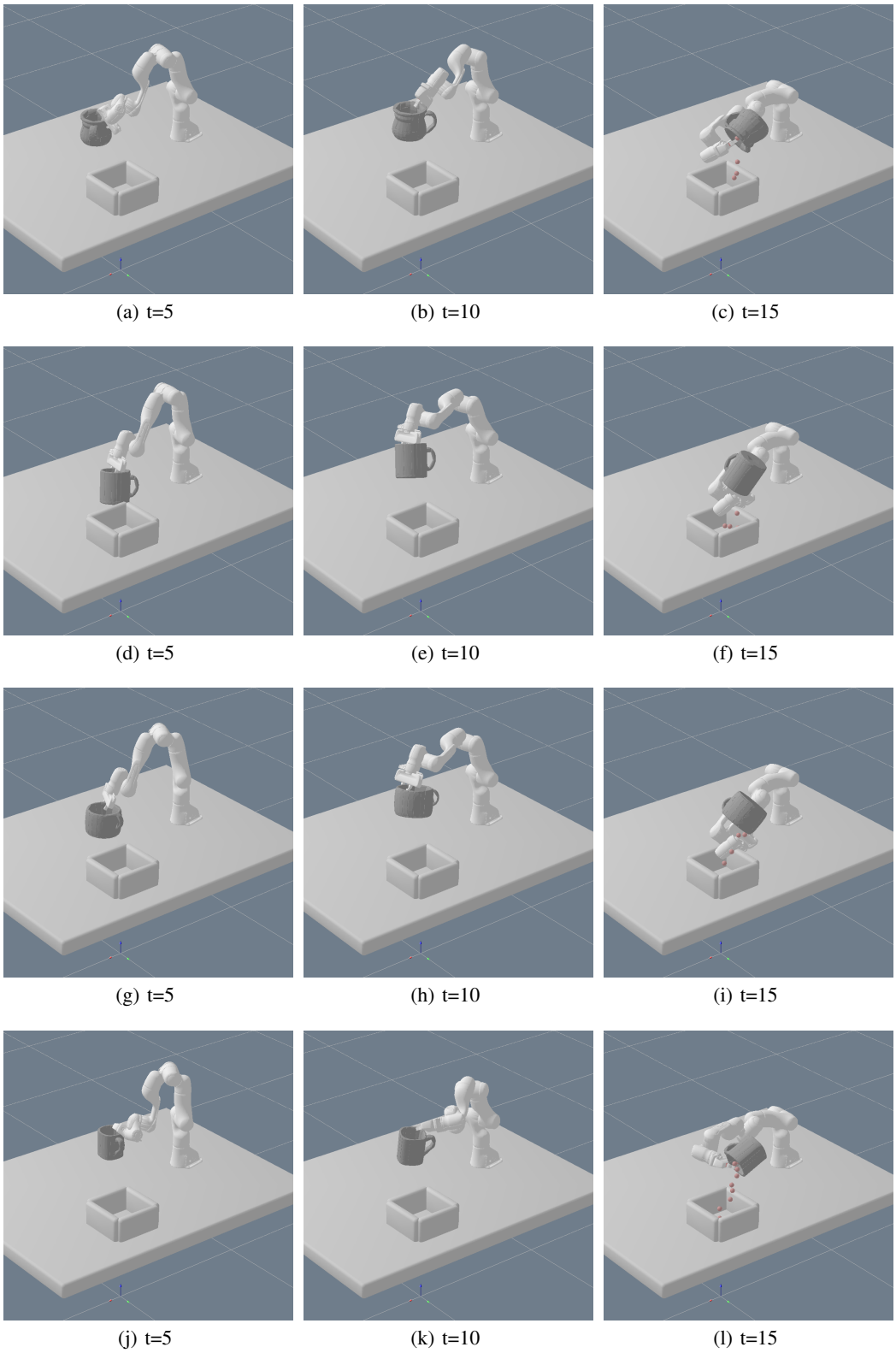
(d) t=13.5 (e) t=15

Figure 12: The handover scenario



(a) t=5 (b) t=10 (c) t=15

Figure 13: Zero-shot Imitation - reference



(a) t=5 (b) t=10 (c) t=15

(d) t=5 (e) t=10 (f) t=15

(g) t=5 (h) t=10 (i) t=15

(j) t=5 (k) t=10 (l) t=15

Figure 14: Zero-shot imitation - optimized motions