Ballpit-NDF: Few-Shot Pouring with Local Neural Descriptor Fields

1st Brian Lee
CSAIL
MIT
Cambridge, MA
brianjsl@mit.edu

2nd Eric Chen
CSAIL
MIT
Cambridge, MA
eric25@mit.edu

Abstract*—Pouring is one of the most commonly executed household manipulation tasks in our daily lives. A successful pour, however, requires both a successful grasp of the surrounding container and successful pouring dynamics on a wide variety of object shapes and configurations. Can a robot robustly perform pouring on objects outside its demonstration distribution and in novel poses? We present Ballpit-NDF—an end-to-end robotic manipulation system capable of pouring balls from novel container shapes using Local Neural Descriptor Fields [2]. Our approach utilizes an imitation learning paradigm that leverages neural implicit representations trained on the surrogate task of object reconstruction to find suitable grasp poses through an energy minimization scheme on *partial* point clouds. In particular, we make use of a limited number of demonstrations of grasping on *out-of-distribution* categories and show generalization performance to novel objects at evaluation time. Combined with a *promptable* perception system that utilizes the Grounding DINO [12] and Segment Anything (SAM) [11] algorithms for segmentation to generate the point clouds, along with off-the-shelf Differential Inverse Kinematics Solvers, we find that our robot can successfully pour a ball into a ballpit 75% of the time for novel objects in three distinct categories given only demonstrations on mug handles. We use a custom Franka Panda robot in a simulated physics environment using the Drake [21] robotics toolbox. Our code is publicly available at https://github.com/brianjsl/BallPitNDF.

I. Introduction

A robot's ability to generalize to new environments and novel objects for manipulation is crucial for determining its performance in the real world. A key challenge in training robotic manipulation systems is the limited number of available demonstrations, which may not represent the types of objects encountered at test time. For example, can a robot that has been taught to manipulate mugs also be adapted to manipulate objects of new categories such as baskets?

In our paper, we implement an end-to-end robotic manipulation system that builds on top of the Local Neural Descriptor Field [2] framework to select a grasp pose for objects of various novel classes including baskets, bowls, and new classes of mugs, for a commonly executed household manipulation task: pouring. A succesful *general* pouring system, however, requires a succesful grasping mechanism that generalizes across categories.

Our system, dubbed BallpitNDF, works by applying *promptable* deep geometric perception using a combination of the newly proposed Grounding DINO [12] and Segment Anything [11] algorithms with outlier pruning to segment out the objects given text prompts before creating a point cloud. Using the Local Neural descriptors of poses given from a set of baselines demonstrations on out-of-distribution object categories (say

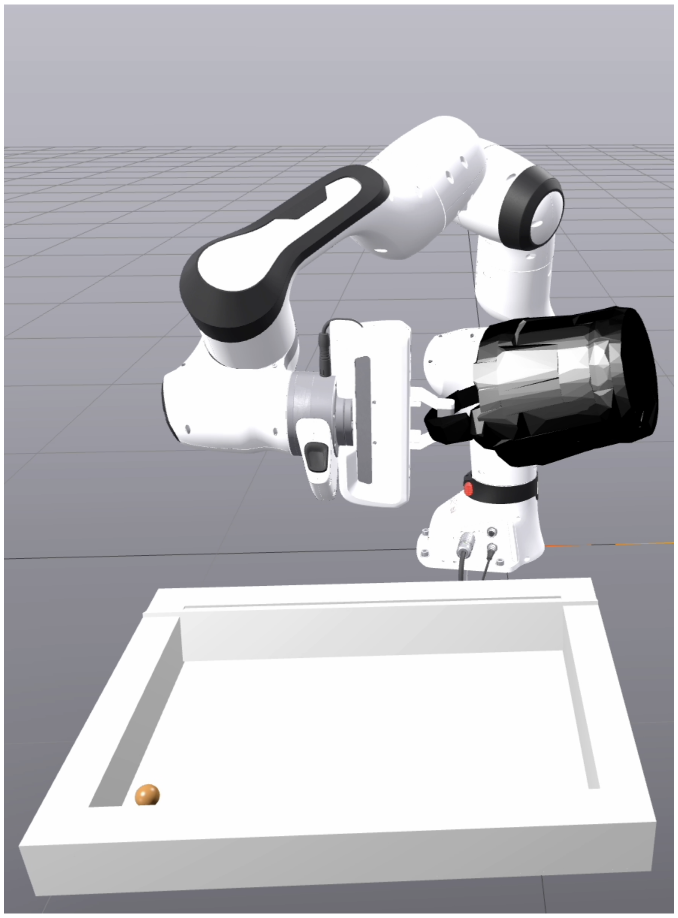


Fig. 1: Using a three stage pipeline consisting of a deep perception system using Grounding SAM, an energy-minimization scheme to generate grasps using LocalNDF descriptors, and off-the-shelf Differential IK solvers, we find that our robot can succesfully pour a ball into a ballpit for a wide array of novel object types 75% of the time.

mug handles), our model then utilizes an energy optimization with Local Neural Descriptors to generate new poses before using off-the-shelf differential inverse-kinematics solvers for the final pouring task. Our preliminary testing shows that BallPitNDF is capable of pouring a ball into a ballpit succesfully on a diverse array of novel object categories in arbitrary SE(3) poses.

II. Related Work

A. Mask Generation with Deep Perception

When generating point clouds with registered cameras on objects, we often need to mask or segment out the desired objects of interests. A vast array of literature exists [14] on the topic, most notably the simple MASK R-CNN approach used by He et. al [7]. Mask R-CNN, however, suffers from the limitaiton that it must be fine-tuned on all object categories of interest. In contrast, a recent approach called Grounding DINO [22] makes use of contrastive learning to achieve a zero-shot generalization capability to new labels. Grounding DINO was trained on approximately 10 million images, extracting image and text features to create a unified representation. Building on Grounding DINO, Grounding SAM [16] leverages the bounding boxes returned by Grounding DINO as prompts to Segment Anything [11] for generating segmentation masks. Trained on over 1 billion masks and 11 million images, the Segment Anything model consists of a prompt encoder, image encoder, and mask decoder to predict segmentation masks given a prompt such as image coordinates or text. The out-of-the-box bounding and segmentation capabilities of both modules on novel object categories through the prompting mechanism make them appealing choices for our own perception tasks.

B. Pouring

Pouring is a general manipulation task that holds great potential for streamlining various household tasks including beverage preparation and object transferring. Previous approaches [8] have utilized self-supervised training, but still suffer from the fact that they only work on objects seen at training time and in default poses. Our approach works more generally, for objects in novel poses and novel categories.

C. Generalizable Manipulation

Various imitation learning pipelines have been used for robotic manipulation tasks. For known objects, pose estimation can be used for manipulation [18], while for novel object shapes, template matching approaches with coarse 3D primitives can be used [6]. Unfortunately, these approaches often fail when presented with objects that differ substantially from simple geometric primitives. Our approach works by capturing more general structure within object shapes.

D. Neural Implicit Representations for Robotic Manipulation

Neural implicit representations [13], [15] have emerged as promising 3D representations for various robotic manipulation tasks. Previous approaches have made use of Neural implicit representations for a variety of manipulation tasks [9], [10], [17], [19]. In particular, the work of [2], [5], [17], [20] build on the Neural Descriptor Field framework [19] for learning manipulation skills in a few-shot manner, where underlying high-dimensional neural descriptors are used to transfer and generalize demonstrations to novel object poses. Our work utilizes the approach of [2] which extends Neural Descriptor Fields to objects in not only novel *poses* but in novel *categories* as well.

III. Background: Local Neural Descriptor Fields

A. Neural Descriptor Fields

Neural Descriptor Fields (NDF) are a form of learned object representations that enable robots to perform manipulation tasks on objects with arbitrary SE(3) pose from a small number of demonstrations. The representation is given by a continuous function $f(\mathbf{x}|\mathbf{P}) : \mathbf{R}^3 \times \mathbf{R}^N \rightarrow \mathbf{R}^d$ that encodes a 3d coordinate $\mathbf{x}$ and point cloud $\mathbf{P}$ into a spatial descriptor. These latents capture the relationship between the point and the object geometry such that instances of a category of objects (ie. mugs) would have similar neural descriptor values at geometrically similar features (ie. mug handles).

NDFs are trained in a fully self-supervised manner using the surrogate task of object reconstruction. More specifically, NDFs train an occupancy neural network as proposed by Mescheder et. al [13]. The occupancy network $\Phi(\mathbf{x}, \mathcal{E}(\mathbf{P}))$, predicts the occupancy value (1 if inside the object, 0 otherwise) given a 3D point $\mathbf{x}$ and an encoded point cloud $\mathcal{E}(\mathbf{P})$ from PointNet. After training, the intermediate activations are concatenated into a vector to get the neural descriptor field.

$$f(\mathbf{x}|\mathbf{P}) = \bigoplus_{i=1}^{L} \Phi^i(\mathbf{x}, \mathcal{E}(\mathbf{P})) \tag{1}$$

Neural Descriptor Fields can also be extended from individual point descriptors to descriptors representing full SE(3) poses. This is achieved by associating descriptors with a rigid set of $n \geq 3$ non-collinear query points, $\mathcal{X} \in \mathbf{R}^{3\times N_q}$, that are constrained to transform rigidly together. The SE(3) pose $\mathbf{T}$ is represented by its action on these points, $\mathbf{T}\mathcal{X}$, and the category-level pose descriptor $\mathbf{Z}$ is computed as:

$$\mathbf{Z} = F(\mathbf{T}|\mathbf{P}) = \bigoplus_{\mathbf{x}_i \in \mathcal{X}} f(\mathbf{T}\mathbf{x}_i|\mathbf{P}), \tag{2}$$

where F maps the point cloud $\mathbf{P}$ and the SE(3) pose $\mathbf{T}$ to $\mathbf{Z} \in \mathbf{R}^{d\times N_q}$, enabling consistent encoding across similar object categories.

NDFs facilitate few-shot learning of manipulation tasks using K demonstrations $\{D_i\}_{i=1}^K$, where each $D_i = (\mathbf{P}^i, \mathbf{T}^i_{\text{grasp}})$ consists of an object point cloud $\mathbf{P}^i$, and a grasp pose $\mathbf{T}^i_{\text{grasp}}$. The query points $\mathcal{X}_{\text{grasp}}$ are used to encode spatial descriptors $\{(\mathbf{Z}^i_{\text{grasp}})\}_{i=1}^K$ for each pose:

$$\mathbf{Z}^i_{grasp} = F(\mathbf{T}^i_{grasp}|\mathbf{P}^i) \tag{3}$$

These descriptors are averaged across demonstrations to obtain category-level descriptors $\bar{\mathbf{Z}}_{\text{grasp}}$. At test time, a novel object

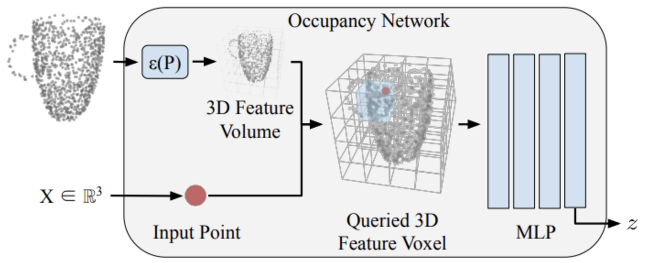


Fig. 2: A L-NDF takes a coordinate $\mathbf{x}$ and a conditioning point cloud $\mathbf{P}$ and uses an encode ε to encode P into a 3D feature volume from which the voxel containing x is queried. These feature are passed into an MLP where the final activations are extracted to create a point descriptor

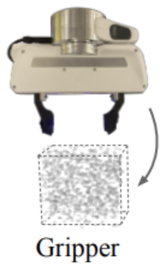


Fig. 3: Query Point initialization for the L-NDF

point cloud $\mathbf{P}_{\text{test}}$ is used to predict the SE(3) poses $\mathbf{T}^{\text{grasp}}_{\text{pick}}$ through an energy minimization objective:

$$\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \|F(\mathbf{T}|\mathbf{P}) - F(\hat{\mathbf{T}}|\hat{\mathbf{P}})\|. \tag{4}$$

where the target pose $\hat{\mathbf{T}}$ is found by optimizing over the object pose 1 landscape to find a matching pose descriptor.

A desirable property of any category-level descriptor is that of SE(3) equivariance: that is, the property that the descriptors remain invariant under rigid transformations, that is, for any rigid transform $\mathbf{R} \in \text{SE}(3)$,

$$f(\mathbf{x}|\mathbf{P}) \equiv f(\mathbf{Rx}|\mathbf{RP}) \tag{5}$$

NDFs as proposed by Du et. al [19] achieve this through specialized design choices in the form of Vector Neurons [4].

B. Local Neural Descriptor Fields

Despite excelling at handling arbitary poses within a category, a key limitation of Neural Descriptor Fields (NDFs) is the fact that they are merely *category-level* descriptors and are thus unable to generalize to new object classes. This limitation is primarily due to the PointNet encoder which produces a global latent for the entire object. General objects, however, often share common geometric shapes or features, which motivates the usage of *local* features to generate more generalizable features.

Local Neural Descriptor Fields (L-NDFs) as proposed by Chun et. al [2] circumvents some of these limitations by encoding object geometry locally using a latent grid of voxels (see Fig. 2). This improved encoding scheme allows L-NDF to produce better object representations for unseen classes at test time. Unlike the approach of Du et. al, however, LocalNDFs do not have SE(3) equivariance built-in. This is remedied by adding a contrastive loss term that enforces descriptor similarity between objects. More specifically, this is done by enforcing descriptor similarity under a rigid transform $\mathbf{T}$ to be roughly proportional to their inverse distance across different rigid transformations:

$$\text{sim}(f(\mathbf{x}_1|\mathbf{P}), f(\mathbf{Tx}_1|\mathbf{TP}) \propto \frac{1}{\|\mathbf{x}_1 - \mathbf{x}_2\| + \epsilon} \tag{6}$$

where ϵ is some stability constant. This is done by using a target reference point and comparing the cosine similarity of the reference point with the inverse distance of a set of $k-1$ other sample points. See [2] for additional details.

For few-shot manipulation tasks, L-NDFs leverage the voxelized encoding scheme to optimize object poses by focusing on the localized geometric features. Unlike NDFs, which use a global optimization of the query point locations initialized at a random orientation, because L-NDFs only aggregate information across local geometry,there is little information relating distant geometric features. L-NDFs thus use a random initial translation to transform the query points within the bounding box of the observed point cloud to ensure proximity to the target feature for accurate pose optimization.

Query point selection is crucial for balancing the capture of sufficient local geometry while avoiding irrelevant features or empty space. For precise tasks, such as grasping, query points are sampled near contact geometry, while general tasks, like surface placement, benefit from larger query point clouds that maximize object volume and minimize empty space.

In our project, we use the same query point initialization scheme as in [2], using a rectangle of size similar to the contact geometry of the gripper (see 3).

IV. Methods

Ballpit-NDF consists of 3 main modules:

1) **Perception**: We use a deep perception system that uses GroundingDINO [12] and Segment Anything [11] to crop out the target object in each camera before returning a point cloud. Our deep perception systems are *promptable*, allowing for bounding and segmentation to get target point clouds despite *never having seen* the target object class.
2) **Grasping**: We perform an energy minimization using the LocalNDF descriptors to get target grasps from a set of out-of-distribution demonstrations. This grasping is SE(3) equivariant and generalizable to even out of demonstration distribution object classes.
3) **Kinematics**: We use a simple trajectory planner that utilizes off the shelf diff-IK with a state machine to create a grasp and pour trajectory.

The primary contributions of our work are twofold: first, the addition of a generalizable, robust deep perception module

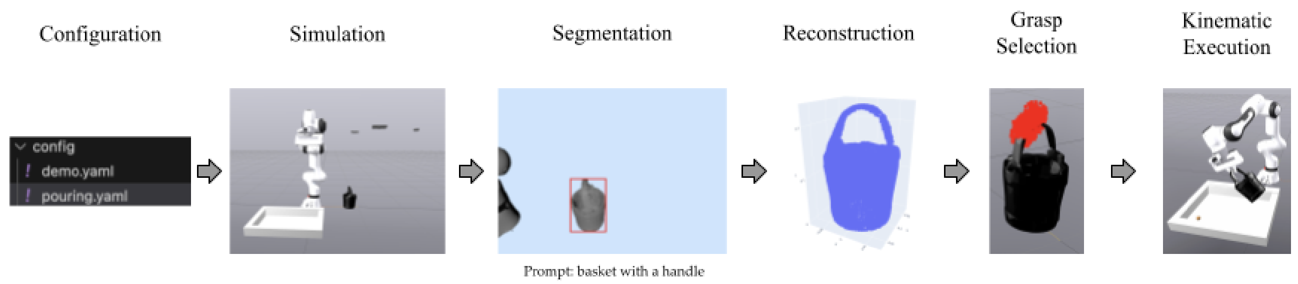


Fig. 4: **System Overview**: Our simulation consists of three main modules: a perception module that utilizes GroundedSAM to get object point clouds, a grasping module that does an energy minimization with the LocalNDF to generate target grasps, and a Diff-IK planner used to execute the pouring. The target prompt (name of the object class) given to the GroundedDINO/SAM module, along with query point hyperparameters used for simulation can be tuned with the hydra configuration file.

allows the system to be completely *full-stack*. In contrast to previous works using Neural Implicits [2], [19], our system is also designed to work on the more difficult task of pouring, rather than simple pick and place.

A. Environment Setup

We use the Drake [21] robotics framework as the simulation environment for our robotic experiments.

For all our experiments, we use a custom simulated Franka Panda robot equipped with a Franka Hand Gripper, fixed at the origin $(0, 0, 0)$, with an `InverseDynamicsDriver`. The authors chose a Franka Panda Arm (along with Panda Hand) as the authors were interested in reintegrating some of the code used in the original L-NDF paper for creating a demo station which was simulated in Pybullet [3] (although we ended up rewriting the demo station from scratch in Drake) along with the possibility of testing on Real Franka Panda Arms that we had available.

A novel container (typically a basket but we also test with other objects including bowls, and mugs in novel poses) is placed at position $(0.5, 0, 0)$. A set of N (where N is a customizable configuration parameter) balls are dropped from random heights into the container. We typically set $N = 1$ for all experiments due to the speed of the physics engine with a larger number of balls. We find from testing, however, that the system can genaralize to an arbitrary number of balls as well (tested up to $N = 7$). A ball pit (simulated using a standard `manipulation` bin) is placed on the floor at position $(0, -0.2, 0)$ (although the exact position of the board is a configurable parameter). We use balls as the object to be poured mainly as a substitute for other things we might pour in our day to day lives such as fluids due to the fact that fluids are not yet available in the Drakes physics simulator.

We modeled a custom basket using Blender and used object meshes of spheres we found on the internet for the balls from the internet before processing them into `sdf` files. The spheres use a simple point-contact model as they are simple geometric primitives, while all other objects are constructed to use hydroelastic contact with the `make_sdf()` function

TABLE I: Prompts for the Perception Module

Object Class	Prompt
`mug`	`a coffee mug with a handle`
`basket`	`a basket with a handle`
`bowl`	`a bowl with a handle`

in the `manipulation` package. The other objects (mugs, bowls, and so on) are slightly modified from those publicly available in the Shapenet [1] dataset. All objects are created with coefficients of friction $\mu_{static} = 1.2$ and $\mu_{dynamic} = 0.8$ with hydroelastic modulus $5e^7$ and Hunter-Crossley Dissipation 1.25.

To capture the object from multiple perspectives, three RGBD cameras are positioned around the object. No "cheat" cameras are positioned underneath the object and we use only the partial point cloud for the L-NDF energy minimization.

All configurable parameters (eg. L-NDF model size, optimizers, query point initialization, object scale, and so on) can be configured through a modifiable Hydra config file.

B. Perception

Our perception system takes in the RGB-D sensor readings from 3 RGB-D sensor cameras and concatenates them to return a merged partial point cloud of the container. The initial sensor reading includes extraneous background objects (such as the arm and other sensors), so preprocessing is required to isolate the container. The high-level approach is to identify where the container is and then crop it out before generating the point cloud.

Our approach first gets the bounding box of the basket in the RGBD images using the Grounding DINO model from Huggingface [12]. We prompt the detection model with the prompts given in table I and set the box and text threshold to 0.3. The bounding boxes are passed into Grounded Segment Anything [16] to retrieve the segmentation masks for the basket. To get proposal crop coordinates for the basket, we

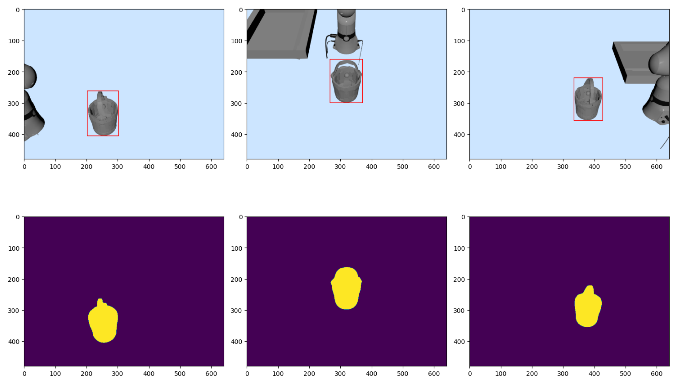


Fig. 5: Bounding boxes for the target object, the basket, from Grounding DINO, and the segmentation masks from Grounding SAM. These results help generate crop coordinates for the point cloud.

project the 2d coordinates of the bounding box (XYXY) to the camera frame by undoing the pinhole projection.

$$X_c = (u - c_x)\frac{Z_c}{f_x} \tag{7}$$

$$Y_c = (v - c_y)\frac{Z_c}{f_y} \tag{8}$$

However, naively doing this doesn't work because the Z coordinates of the bounding box's top left and bottom left coordinates are ambiguous. In many cases, they return `inf` or are in conflict with objects in the background.

To address this issue, we retrieve the depth map within the segmentation mask and extract the minimum and maximum finite depth values. For each bounding box point, we generate two 3D coordinates: one closer to the camera and one further away using the min and max depth values above to mitigate the ambiguity in the z-axis. We concatenate all the bounding box 3d coordinates then compute the lower left and upper left coordinate. To make the bounding boxes robust to slight errors near the boundaries of the segmentation mask, we utilize Statistical Outlier Removal as shown in 8: more specifically, we use the `KDTree` library for efficient k- nearest neighbor search for each point and then remove all points with mean distance for that point with the nearest neighbors above 1.5 standard deviations for the global k-NN distances. The point cloud from each camera is then cropped and merged to form the partial point cloud of the basket. Finally, we connect the point cloud output as input to the grasping module to select a grasp pose for the basket.

C. Grasping with LocalNDFs

Our system makes use of the energy minimization scheme proposed in [2] and explained in Section IIIB.

We use a pre-trained Occupancy Network with latent dimension 128 to get the Neural Descriptors, and collect demos of handle grasping using the Drake environment with a manual controller for the Panda Hand. For our query point initialization we use 20 random initial query points, along with 500

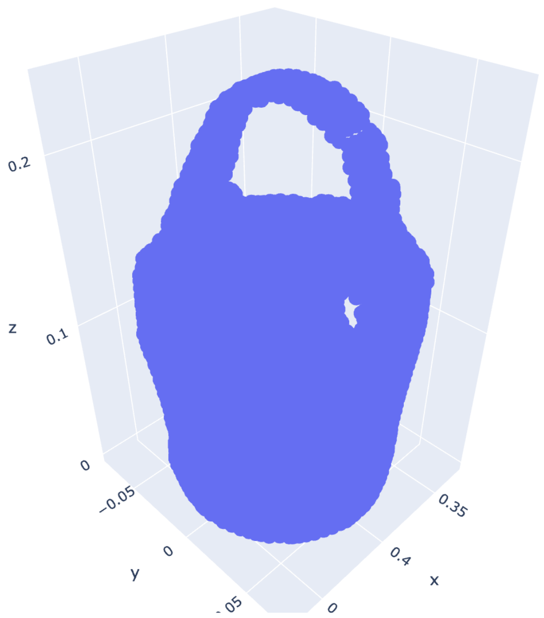


Fig. 6: Partial point cloud of the basket obtained by cropping and merging the point cloud from each camera.

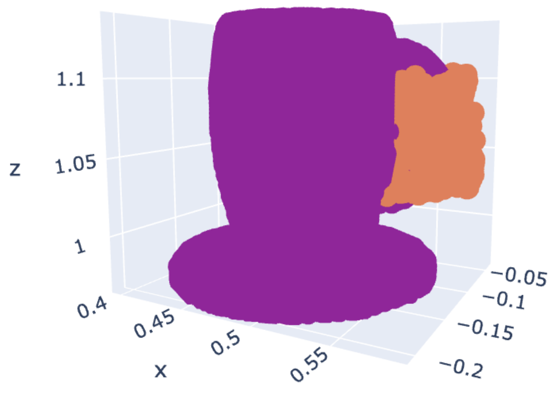


Fig. 7: We give the robot 10 different demonstrations of grasping mug handles and take the average of the neural descriptors across demonstrations as described in Section IIIA.

optimizer iterations. We give the robot 10 demonstrations of grasping on a set of different Shapenet [1] mugs as shown in Figure 7.

We then do an energy minimization to get the locations of the handle query points on our new object as described in IIIB along with the corresponding grasp pose. Because the grasp pose used for the LocalNDF descriptors describes the pose of the end-effectors rather than the entire robot hand, when actually grasping the handle of the object in question we give

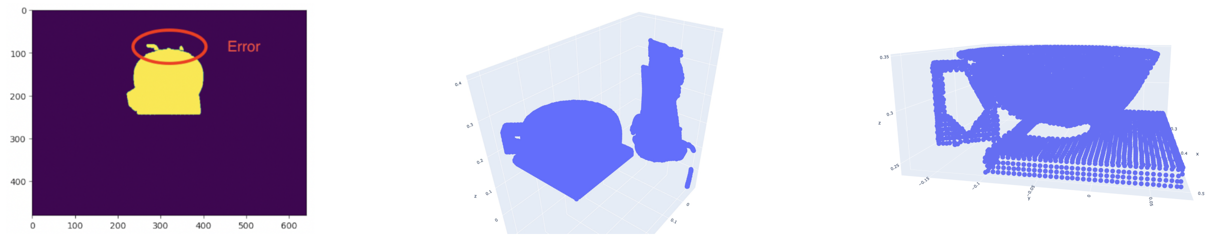


Fig. 8: Errors in the edges of the segmentation map (left) can lead to failures in the unprojected point cloud (middle). We thus use statistical outlier removal to remove outliers from our perception maps to get our crops (right)

the robot hand an offset of $(0, 0, -0.1)$.

D. Kinematics

We use Drake's Differential Inverse Kinematics Solver to produce viable trajectories for the Panda Arm with a simple state-machine planner that moves between 8 different states as illustrated in Figure 7. The `UNPOUR` state (as shown in the figure) refers to the state of waiting for the bucket to get back to level.

V. Evaluation

A. Pouring Results

To quantitatively measure the success of our pouring system, we tested the percentage of times our Robot successfully poured the ball into the ballpit across a range of objects in three different novel categories (i.e. not seen at demonstration time): namely, a basket, a bowl, and a mug not seen during the demonstrations in *arbitrary* rotations about the z-axis (due to the need to put the ball in the container, the containers had to be upright). The balls were dropped at a random height above the container. We did the testing by measuring whether or not the link of the `sphere` (ball) object was within the geometry of the ballpit at the end of each physics simulator where we run each simulation for 40 seconds (roughly the amount of time needed for the ball to get into the pit). We summarize our results in Table II. All object classes are tested for 100 iterations each.

B. Analysis

As expected, the system performs best with the `mug` class with a 94% success rate as the test distribution matches the demo distribution of mug handles. This is likely due to the fact that the handles of the test mugs are similar to those of the demonstration handles, despite the overall mugs often being

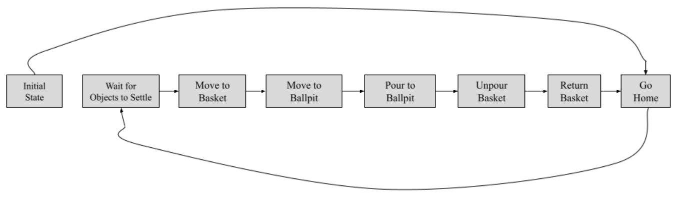


Fig. 9: State Machine Used for Trajectory Planning

significantly different from the test mugs. The system performs second-best on the `basket` class with a 83% success rate, also likely because the handle geometry is quite similar to the demonstration mugs. Conversely, the model performs rather poorly on the `bowl` class with only a 48% success rate due to the differing geometry between test and demo shape distributions.

TABLE II: Pouring Results with Mug Handle Demos

Object Class	Iterations	Pouring Success Rate (%)
`mug`	100	94
`basket`	100	83
`bowl`	100	48
average	300	75

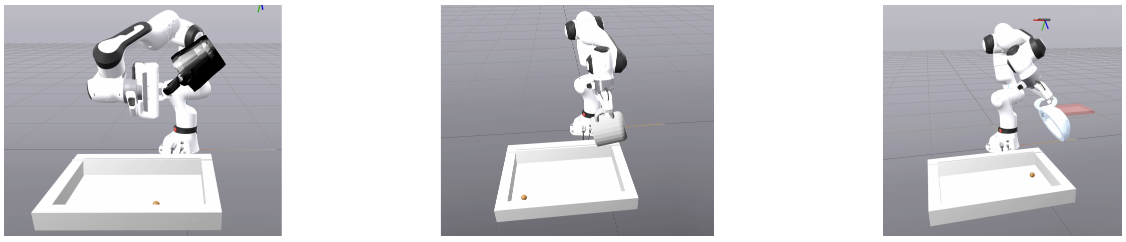
Due to the random query point initializations of the LNDF, along with other possible modes of failures (eg. collision errors) the gripper occasionally fails to correctly grasp the bucket. We discuss more limitations in section C.

We also try giving the robot demonstrations of grasping the rims of a bowl and we find (from a few experiments) that the robot is able to succesfully grab the rim of the basket (rather than the handle) and succesfully pour, although we find that the success rate of pouring the ball into the bin is not as high as when gripping the handle.

C. Limitations and Next Steps

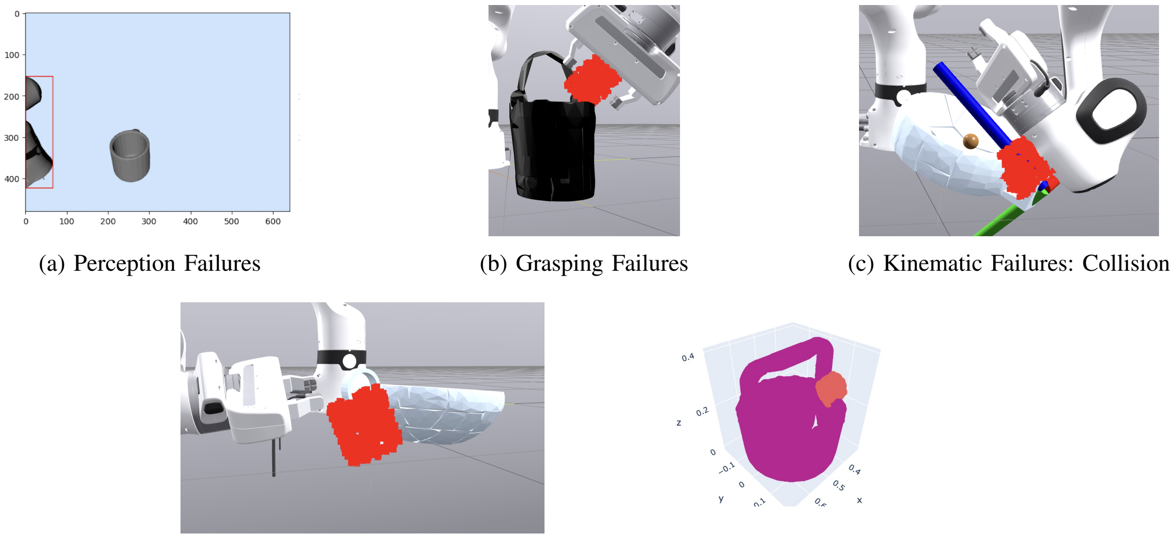
Perception Failures: Because our model makes use of promptable vision models for segmentation, the model can give a poor crop without a suitable prompt. For example, in Fig 11 we use the prompt `a basket` rather than `a basket with a handle` and we get a poor segmentation leading to inaccurate point clouds. Similarly, we get poor results when just using a `mug` so we use the more descriptive `A coffee mug with a handle`.

Grasping Failures: Due to the random translational/rotation initialization of the query points, on rare occasions the energy minimization provides a bad grasp that leads the robot to be unable to grab the basket. This can largely be mitigated by running multiple random initializations (we use 20) and taking the best one, but on rare occasions a grasping failure is unavoidable with the current approach. Another possible approach worth pursuing in the future is using the perception mechanism to create a "rough" intialization of the query points



(a) A basket (b) A mug of new class (c) A Bowl

Fig. 10: Our object is capable of pouring objects of a diverse array of (out of distribution) classes, including baskets, new mugs, and bowls



(a) Perception Failures (b) Grasping Failures (c) Kinematic Failures: Collision

(d) Kinematic Failures: Singularities (e) Generalization Failures

Fig. 11: **Failures**: a) Perception Failures due to poor prompting. Here we use the prompt `a mug` rather than the more descriptive `a coffee mug with a handle`. b) A grasping failure due to poor stochastic initializations of the query points. c) A kinematic failure has resulted due to collision between the object geometry and the gripper during the grasping phase. d) A kinematic failure has occured due to the gripper getting into a singularity in the Diff-IK e) Generalization Failures on objects with suitably different geometries.

and then optimizing using the energy minimization scheme to the correct query point position and pose.

Brittleness and Model Generalization: Although our model works well for distinct shapes that are similar to the demonstration shapes, the model still fails to generalize to completely general object classes. We also find from experiment that the sucess rate of the grasps given by the energy minimization procedure drops significantly when the model geometry differs too much from the original demonstrations even though the shapes are of the same class (eg. handles of a basket with a much larger handle, see above).

Kinematic Failures: Like the grasping failures, because we only use off-the-shelf Differential IK solvers, our robot sometimes bumps into parts of the handle while trying to grab the handle. Other times, the robot hand has to extend too far and ends up reaching joint extrema (ie. singularities in the diff-IK), which also leads to kinematic failure. Our work also suffers from having to define a set of keypoints manually for the differential IK solvers. Future directions for this work may include adding motion planning and collision avoidance into the system, either using GCS approaches like IRIS and/or reinforcement learning-based methods (the authors were actually considering doing this but ran out of time).

Computation and Speed: Due to the need to perform on-demand inference on a large vision model (GroundedSAM) along with on-demand energy optimization to get the grasp poses, BallpitNDF is rather slow, with predictions taking up to 25 seconds on average on a Macbook M4 Pro. Parallelizing the optimization with GPU acceleration may help the speed, as may smaller, more efficient bounding box models, although they may not have the same off-the-shelf querying capabilities as our current model.

VI. Conclusion

We introduce BallpitNDF, a full-stack, deep-perception, robotic manipulation system that utilizes a few-shot imitation learning framework to pour balls from novel objects into a ballpit. Our system has been shown to successfully pour from various different novel objects including baskets, bowls, and arbitrary (out of demo distribution) mugs, in arbitrary SE(3) poses using only a small number of demonstrations on mug handles. We also acknowledge the limitations of BallpitNDF, including grasping failures due to the stochastic optimization process, kinematic failures due to occluded pieces, and slow speed due to on-demand inference and optimization in our pipeline. We propose that future works remedy the kinematic issues by adding motion-planning modules and the speed issues with better hardware acceleration. We hope that this work provides a valuable stepping stone in the pursuit of future generalizable full-stack pouring systems. To that end, we have released our code publicly on our github repo for others to build off.

Acknowledgement

We would like to thank the amazing staff of 6.4210 for their wonderful and informative lectures throughout the semester. We would especially like to thank our group's project TA, Ethan Yang, who was especially helpful during OHs and the author of the LocalNDF paper Ethan Chun who helped discuss some of his design choices and whose helpful discussions were critical in the final implementation.

Work Distribution

Brian led the system design, implementation, integration, and developed the LNDF grasping pipeline, camera and point cloud setup, simulation physics on new objects, and kinematic motion planners. Eric led the development of the Panda driver setup, inverse kinematics, deep perception modules (Grounding DINO and SAM), and the experiment testbed. All members contributed to the communication of results.

References

[1] 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, 2015.

[2] Ethan Chun, Yilun Du, Anthony Simeonov, Tomas Lozano-Perez, and Leslie Kaelbling. Local neural descriptor fields: Locally conditioned object representations for manipulation, 2023.

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

[4] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector neurons: A general framework for so(3)-equivariant networks, 2021.

[5] Jiahui Fu, Yilun Du, Kurran Singh, Joshua B. Tenenbaum, and John J. Leonard. Robust change detection based on neural descriptor fields, 2022.

[6] Kensuke Harada, Kazuyuki Nagata, Tokuo Tsuji, Natsuki Yamanobe, Akira Nakamura, and Yoshihiro Kawai. Probabilistic approach for object bin picking approximated by cylinders. In *2013 IEEE International Conference on Robotics and Automation*, pages 3742–3747, 2013.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[8] Yongqiang Huang, Juan Wilches, and Yu Sun. Robot gaining accurate pouring skills through self-supervised learning and generalization, 2020.

[9] Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *CoRR*, abs/2110.14217, 2021.

[10] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. *CoRR*, abs/2104.01542, 2021.

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[12] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2024.

[13] 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 (CVPR)*, June 2019.

[14] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[15] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019.

[16] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[17] Hyunwoo Ryu, Hong in Lee, Jeong-Hoon Lee, and Jongeun Choi. Equivariant descriptor fields: Se(3)-equivariant energy-based models for end-to-end visual robotic manipulation learning, 2023.

[18] John Schulman, Jonathan Ho, Cameron Lee, and P. Abbeel. Learning from demonstrations through the use of non-rigid registration. In *International Symposium of Robotics Research*, 2013.

[19] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. *CoRR*, abs/2112.05124, 2021.

[20] Anthony Simeonov, Yilun Du, Lin Yen-Chen, Alberto Rodriguez, Leslie Pack Kaelbling, Tomas Lozano-Perez, and Pulkit Agrawal. Se(3)-equivariant relational rearrangement with neural descriptor fields, 2022.

[21] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[22] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022.