Distilled Feature Fields Enable Few-Shot Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Self-supervised and language-supervised image models contain rich knowledge of the world that is important for generalization. Many robotic tasks, however, require a detailed understanding of 3D geometry, which is often lacking in 2D image features. This work bridges this 2D-to-3D gap for robotic manipulation by leveraging distilled feature fields to combine accurate 3D geometry with rich semantics from 2D foundation models. We present a few-shot learning method for 6-DOF grasping and placing that harnesses these strong spatial and semantic priors to achieve in-the-wild generalization to unseen objects. Using features distilled from a vision-language model, CLIP, we present a way to designate novel objects for manipulation via free-text natural language, and demonstrate its ability to generalize to unseen expressions and novel categories of objects.

1 Introduction

What form of scene representation would facilitate open-set generalization for robotic manipulation systems? Consider a warehouse robot trying to fulfill an order by picking up an item from cluttered storage bins filled with other objects. The robot is given a product manifest, which contains text or images it needs to identify the correct item. Geometry plays an equally important role as semantics, since the robot needs to comprehend which parts of the object geometry afford a stable grasp. Undertaking such tasks in unpredictable environments — where items from a diverse set can deviate markedly from the training data, hidden or jumbled amidst clutter, underscores the critical need for robust priors in both spatial and semantic understanding.

In this paper, we study few-shot and language-guided manipulation, where a robot is expected to pick up novel objects according to a few examples or text descriptions without having seen a similar item in the past. Toward this goal, we build our system around pre-trained image embeddings, which have emerged as a reliable way to learn commonsense priors from internet-scale datasets.

Figure 1 shows how our system works. First, the robot scans a tabletop scene by taking a sequence of photos using an RGB camera held on a selfie stick (Figure 1, left). These photos are used to construct a NeRF of the tabletop, which, crucially, is trained to render not just RGB colors but also *image features*. This produces a scene representation that embeds knowledge from 2D pretrained features into a 3D volume (Figure 1, middle). The robot then references demos and language instructions to grasp objects specified by a user (Figure 1, right).

This scene representation is called a Distilled Feature Field (DFF), and was introduced in graphics for tasks such as decomposing and editing images [1, 2]. The main contribution of this work is to study the use of DFFs instead for robot manipulation. We start by experimenting with features from self-supervised vision transformers (DINO ViT, see [3]) that have been shown to make effective out-of-the-box visual descriptors for dense correspondence [4]. We then integrate features from a vision-language model, CLIP [5], which is a strong zero-shot learner on various vision and visual question-answering tasks.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

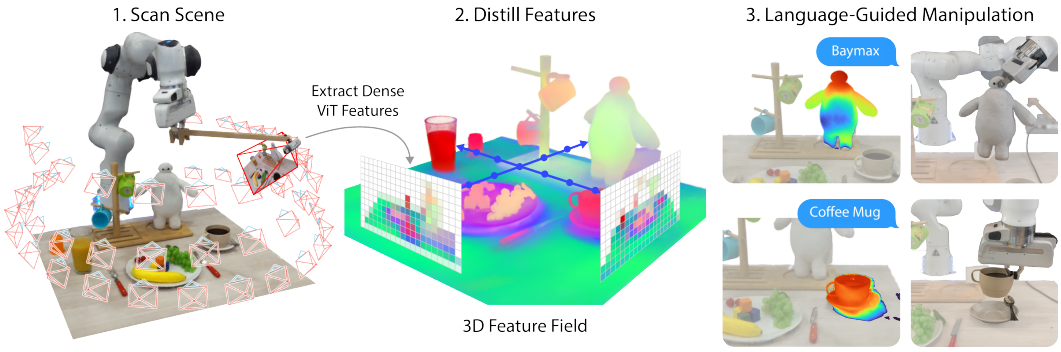


Figure 1: **Distilled Feature Fields Enable Open-Ended Manipulation.** 1) Robot uses a selfie stick to scan RGB images of the scene (camera frustums shown). 2) Extract patch-level dense features for the images using a Vision Transformer (ViT), and distill them into a feature field along with modeling a NeRF (PCA shown). (c) We can query CLIP feature fields with language to generate segmentation heatmaps, and infer 6-DOF grasps on novel objects given only ten demonstrations.

To make using distilled feature fields practical for manipulation, we speed up the creation of the Feature Fields by building upon recent techniques for faster NeRFs [6, 7, 8], and introduce a way to extract patch-level dense image features from CLIP that preserves its alignment with the original language grounding. We present few-shot learning experiments that demonstrate our robot's ability to handle open-set generalization to objects that differ significantly in appearance, materials, and poses. We also present language-guided manipulation experiments and show that our robot can generalize to unseen categories of objects using just ten demonstrations on four object categories.

2 Problem Formulation

We consider the class of manipulation problems that can be parameterized via a single rigid-body transformation $\mathbf{T} \in SE(3)$. Take 6-DOF grasping with a parallel gripper for example (see Figure 2), we can parameterize a grasp as $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ in the world frame. $\mathbf{R}$ is the rotation matrix, and $\mathbf{t}$ is the translation vector. In each scene, the robot is given a set of RGB images $\{\mathbf{I}\}$ and their corresponding camera poses. $\mathbf{T}^*$ is a pose that accomplishes the task.

Few-shot Manipulation. We want to minimize the human effort required to specify an object to manipulate for a generalist robot. *During learning,* the robot is given a few demonstrations. Each demonstration, denoted as D, contains the tuple $\langle \{\mathbf{I}\}, \{\mathbf{T}^*\} \rangle$, where the robot has sampling access to renderings of the scene via N RGB images $\{\mathbf{I}\}_{i=1}^{N}$. *During testing,* the robot is given a new scene, observed by multiple views, $\{\mathbf{I}'\}$. The goal of the robot is to predict a pose $\mathbf{T}$ that solves the task. We consider an in-the-wild setting involving distractor objects and clutter (Figure 6). We are also looking for open-ended generalization: the new scene contains similar but previously unseen objects that differ from the demo in shape, size, pose, and in general, material and appearance.

Open-Text Language-Guided Manipulation. In this problem formulation, we extend few-shot manipulation to include conditioning via open-text in natural language. Given text description of an object, the task is to pick up objects that fit this description. The robot has access to a few demonstrations for multiple object types. At inference time, the robot is given text expressions L^+ to specify which object to operate on. To reject distractors in the scene, it is helpful to also provide negative text L^-, which can be quite general. Note that the object we care about at test time does not have to appear in the demonstrations. We explicitly seek generalization to new object categories using only demonstrations limited to a few types of objects.

3 Open-Ended Manipulation with Feature Fields

Tackling open-ended manipulation using 3D feature fields involves solving three separate problems. First, how to quickly produce the 3D Feature Field of a scene; second, how to represent and infer the

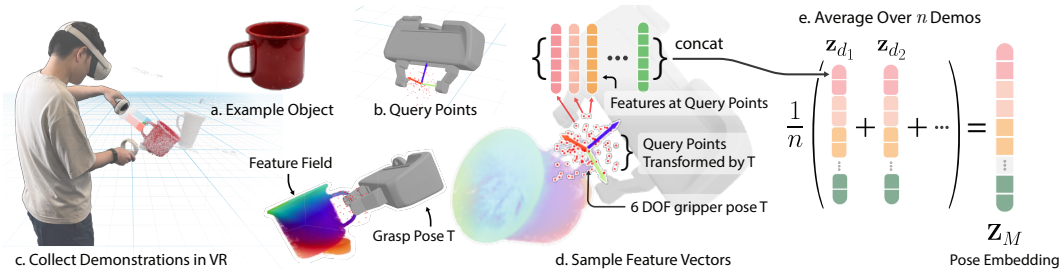


Figure 2: **Representing 6-DOF Poses.** (a) An example mug. (b) Query Points are defined in the gripper's canonical frame. (c) Collect demonstration in a virtual reality environment (VR) by recording the gripper pose $\mathbf{T}^*$. (d) Represent the 6-DOF grasp by the features sampled at the query point cloud, transformed by $\mathbf{T}^*$. Feature vectors are concatenated and averaged over n (in our case $n = 2$) demonstrations. This gives a pose embedding $\mathbf{Z}_M$ for the task M.

rigid-body transformation for 6-DOF manipulation; and finally, how to incorporate language guidance to enable open-text commands. We include a formal summary of NeRFs [9] in Appendix A.1.

3.1 Fast Feature Field Distillation (F3D)

DFFs [1, 2] extend NeRFs by including an additional output to reconstruct dense 2D features from a vision model $\mathbf{f}_{\text{vis}}$. The feature field $\mathbf{f}$ maps a 3D position $\mathbf{x}$ to a feature vector $\mathbf{f}(\mathbf{x})$. We assume that $\mathbf{f}$ does not depend on the viewing direction $\mathbf{d}$[1]. The feature vector corresponding to a camera ray $\mathbf{r}$ is given by the feature rendering integral between the near and far plane (t_n and t_f),

$$\mathbf{F}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}_t)\mathbf{f}(\mathbf{r}_t)\,\mathrm{d}t. \tag{1}$$

Feature Distillation. We begin with a set of N 2D feature maps $\{\mathbf{I}_i^f\}_{i=1}^N$, where $\mathbf{I}_i^f = \mathbf{f}_{\text{vis}}(\mathbf{I}_i)$ for each RGB image $\mathbf{I}_i$. We optimize $\mathbf{f}$ by minimizing the quadratic loss $\mathcal{L}_{\text{feat}} = \sum_{\mathbf{r}\in\mathcal{R}} \left\|\hat{\mathbf{F}}(\mathbf{r}) - \mathbf{F}(\mathbf{r})\right\|_2^2$, where $\mathbf{F}(\mathbf{r})$ is the target feature vector from $\mathbf{I}^f$, and $\hat{\mathbf{F}}(\mathbf{r})$ is estimated by a discrete approximation of the feature rendering integral. Past work [2] jointly optimizes for the RGB and feature reconstruction by minimizing a weighted sum of the photometric loss and the feature loss $\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda\mathcal{L}_{\text{feat}}$. λ is usually set to a small number such as 0.1, and the gradients from $\mathcal{L}_{\text{feat}}$ are usually omitted when updating the density network σ. Existing feature distillation pipelines take up to 20 hours to model a single scene [2, 1]. While this may be acceptable for graphics applications, robots require faster perception systems [12, 13].

We speed up our feature distillation by building off newer NeRF implementations using hierarchical hash grids [6, 8]. We improve upon [2] by implementing a mixed-resolution sampling scheme for the feature maps so that we do not have to up-scale them to match the resolution of the RGB images. This alone significantly reduces the memory footprint, as the high-dimensional features, up-scaled to the RGB image resolution, can require more than 40 GB. In practice, recent implementations use mixed-precision training for better speed, so the larger feature reconstruction loss makes gradient scaling less effective, leading to underflow on the parameters of the RGB field. We offer two implementations in Appendix A.2.1 that both address this problem.

Extracting Dense Visual Features from CLIP. The common practice for extracting dense features from ViTs is to directly use the features from the last layer before pooling. While this approach has yielded great results for vision-only models on image segmentation and image-to-image correspondence tasks [4, 14], it removes the transformations required to project the visual features into the shared feature space with the language stream in vision-language models such as CLIP [5, 15]. Kerr et al. [16] regard CLIP as an image-level feature and construct dense CLIP features by convolving

[1] Prior work that uses the association of image pixels in 3D to supervise learning 2D image features makes the same assumption [10, 11].

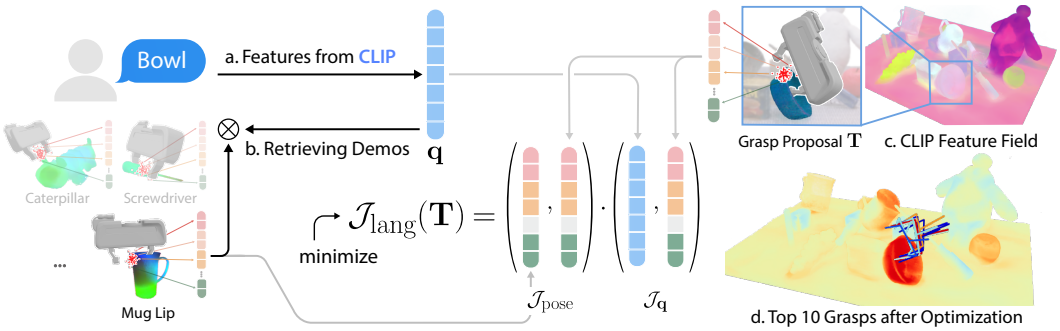


Figure 3: **Pipeline for Language-Guided Manipulation.** (a) Encode the language query with CLIP. (b) Matching Demos: the mug lip demos have the highest similarity to "Bowl". (c) Generate grasp proposals using the CLIP feature field, and optimize them by minimizing $\mathcal{J}_{\text{lang}}$. (d) Top 10 inferred grasps, overlaid on a heatmap showing the similarity between "Bowl" and the CLIP feature field.

the CLIP vision network on multi-scale crops of the RGB image. This procedure involves hundreds of forward-passes through the network and results in blurry feature maps due to the overlap between nearby fields of view. This also makes the features sensitive to scale, causing one to either miss the larger context or essential details.

We present an improved dense feature extraction scheme for CLIP (see Alg.3 in Appendix A.2.2). First, we reparameterize the last layer of the vision network (both the ViT and ResNet) and keep the small output MLP so that the features remain aligned with the language features. We found that removing the residual connection from the class token in Line 5-6 improves the signal-to-noise, at the cost of biasing the features towards local features versus the global one. Second, CLIP uses a small, fixed number of input patches from a square crop. We interpolate the position encoding to accommodate more input patches and arbitrary image aspect ratios [3]. These two techniques enable us to extract patch-level dense 2D features from RGB images at about 25 frames-per-second.

3.2 Representing 6-DOF Poses with Feature Fields

We define a fixed set of N_q query points $\mathcal{X} \in \mathbb{R}^{N_q \times 3}$ in the coordinate frame of the gripper, sampled from a Gaussian. We adjust the covariance and the center of the query points so that it covers the parts of the object we want to grasp (Fig.2b). This allows us to encode a 6-DOF gripper pose $\mathbf{T}$ by transforming the query points $\mathcal{X}$ by $\mathbf{T}$, and *querying* the feature field $\mathbf{f}$ for each transformed point to obtain a local geometric and semantic encoding of the pose (Fig.2d). We query features using the alpha-weighted feature field $\mathbf{f}_\alpha$ to assign less weight to features in free space:

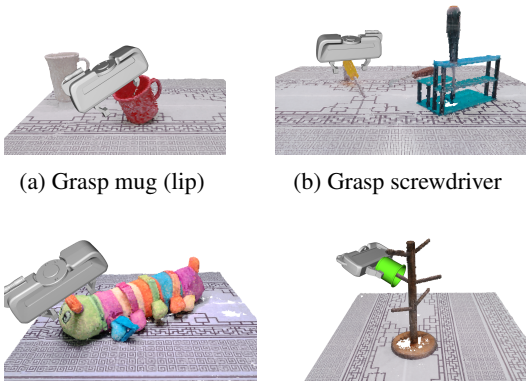
$$\mathbf{f}_\alpha(\mathbf{x}) = \alpha(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}), \tag{2}$$

where $\alpha(\mathbf{x}) = 1 - \exp(-\sigma(\mathbf{x}) \cdot \delta) \in (0, 1)$. σ is the density field from NeRF and δ is the distance between adjacent samples. We encode each demonstration D by concatenating the features sampled at each point in the query point cloud to get $\mathbf{z}_d \in \mathbb{R}^{N_q \cdot f}$, where f is the feature dimensionality. To compute the encoding for a task M, we average the features over the demos to get $\mathbf{Z}_M \in \mathbb{R}^{N_q \cdot f}$ (Fig.2e). During inference, we find the closest corresponding pose by maximizing the similarity over $\mathbf{T}$ via the following cost function.

$$\mathcal{J}_{\text{pose}}(\mathbf{T}) = -\cos(\mathbf{f}_\alpha(\mathbf{T}\mathcal{X}), \mathbf{Z}_M), \tag{3}$$

where $\mathbf{f}_\alpha(\mathbf{T}\mathcal{X})$ are the concatenated features corresponding to the transformed query points for $\mathbf{T}$. This formulation, of matching descriptors for a cloud of query points around a target transform, is similar to the strategy used in Neural descriptor fields [17, 18].

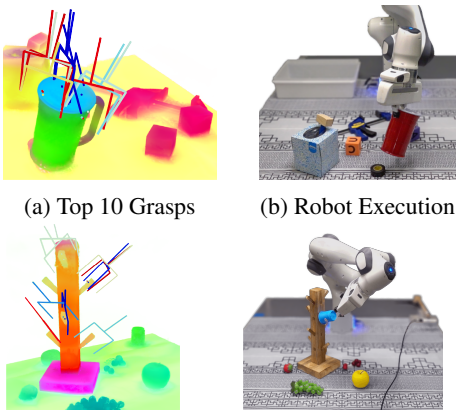
Inferring 6-DOF Poses. To generate the initial set of 6-DOF pose proposals $\mathcal{T} = \{\mathbf{T}_i\}$, we first sample a dense voxel grid over the workspace with voxels of size δ, where each voxel $\mathbf{v}$ represents a translation. We prune irrelevant voxels by first thresholding the alpha values $\alpha(\mathbf{v})$ to mask out free space, and then thresholding the cosine similarity of the voxel features $\mathbf{f}_\alpha(\mathbf{v})$ with the task embedding $\mathbf{Z}_M$. We uniformly sample N_r rotations for each voxel to get $\mathcal{T}$.



(a) Grasp mug (lip) (b) Grasp screwdriver

(c) Grasp caterpillar (d) Place cup on racks

Figure 4: **Five Grasping and Place Tasks.** (a) grasping a mug by its lip or handle (Fig.2); (b) a screwdriver by the handle; (c) the caterpillar by its ears; and (d) placing a cup onto a drying rack. Gripper poses indicate one of two demonstrations.



(a) Top 10 Grasps (b) Robot Execution

(c) Top 10 Grasps (d) Robot Execution.

Figure 5: **Generalizing to Novel Objects.** (Top Row) Mug is much bigger than the ones used for demonstration. (Bottom Row) This rack has shorter pegs with a square cross-section. Demo rack is cylindrical (cf. Fig.4d).

We iteratively optimize the rotation and translation of each gripper pose $\mathbf{T} \in \mathcal{T}$ using the Adam optimizer [19] to minimize Eq. 3. After each optimization step, we prune proposals with the highest loss. Poses with collisions are rejected by thresholding the number of overlapping voxels between a voxelized gripper mesh with NeRF geometry. This leaves us with a ranked list of poses which we input to an inverse kinematics solver and motion planner in PyBullet [20, 21]. We execute the highest-ranked grasp or place pose with a motion plan. The major advantage of our formulation is the ability to infer 6-DOF poses over the entire scene. We have not made any assumptions about objectness, including knowledge of object models, segmentations or poses.

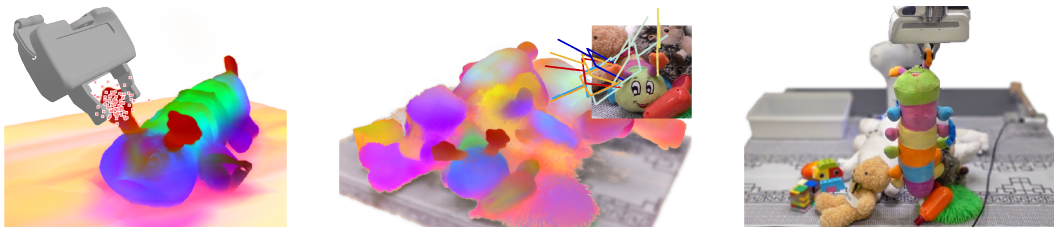
3.3 Open-Text Language-Guided Manipulation

Consider the scene shown in Figure 1. The goal of the robot is to grasp the objects designated by text descriptions. We integrate language guidance into the few-shot manipulation pipeline via the following three steps (see Fig.3): 1) Retrieval of similar demos via query text, 2) initializing grasp proposals, and 3) language-guided grasp pose optimization.

Retrieving Similar Demos via Text. We select the top two demonstrations whose average feature $\mathbf{F}_d$ (averaged among the query points of each example pose $\mathbf{T}_i^*$) is closest to the text embedding $\mathbf{q} = \text{emb}_{\text{CLIP}}(L^+)$. We found that using the positive query text, L^+ alone is sufficient. Formally, this means minimizing the cosine similarity $\cos(\mathbf{F}_d, \mathbf{q})$, which aligns the demonstrations with the query. For "metal mug", "Baymax", "transparent rack", and "measuring beaker", this procedure selects the mug handle, caterpillar, cup on racks, and mug lip demos (Fig.4), respectively.

Initializing Grasp Proposals. Before we can use gradient-based optimization to find the optimal grasp, we first initialize the grasp proposals by filtering out regions in the feature field that are irrelevant to the text query, represented by $\mathbf{q}$. We start by sampling a dense voxel grid among the occupied regions by masking out free space (see Sec.3.2). Afterward, we prune down the number of voxels by keeping those that are more similar to the positive query L^+ than the negative ones L^-. Formally, suppose $\mathbf{q}_i^- = \text{emb}_{\text{CLIP}}(L_i^-) \mid i \in \{1, \ldots, n\}\}$ are the text embedding of the negative queries. We compute the softmax over the pair-wise cosine similarity between the voxel's feature $\mathbf{f}_\alpha(\mathbf{v})$ and the ensemble $[\mathbf{q}, \mathbf{q}_1^-, \mathbf{q}_2^-, \ldots, \mathbf{q}_n^-]$. We thresholded using Otsu's method [22]. Finally, we produce the full poses, $\{\mathbf{T}_i\}$ by uniformly sampling of N_r rotations for each remaining voxel.

Language-Guided Grasp Pose Optimization. We construct the language-guided cost function $\mathcal{J}_{\text{lang}}$ by multiplying the demo-only cost $\mathcal{J}_{\text{pose}}$ from Eq.3 with the language-guidance $\mathcal{J}_{\mathbf{q}}$. This



(a) Demonstration (1 of 2) (b) Feature Field of Cluttered Scene (c) Robot Execution

Figure 6: **Grasping in a Cluttered Scene.** (a) Demonstration for grasping the caterpillar in its DINO feature field (color is PCA, red dots show query points). (b) A cluttered scene with several toys on top of each other. Inset shows the top 10 inferred grasps. Observe the caterpillar's ears share the same features with the demo. (c) Robot successfully grasps the caterpillar.

allows us to assign lower cost to a proposal if the object it targets shares higher similarity with $\mathbf{q}$.

$$\mathcal{J}_{\text{lang}}(\mathbf{T}) = \mathcal{J}_{\text{pose}}(\mathbf{T}) \cdot \mathcal{J}_{\mathbf{q}}(\mathbf{T}), \qquad \mathcal{J}_{\mathbf{q}}(\mathbf{T}) = -\cos\left(\mathbf{F}_{\alpha}(\mathbf{T}\mathcal{X}), \mathbf{q}\right) \tag{4}$$

where $\mathcal{J}_{\mathbf{q}}$ represents the normalized similarity between the CLIP-encoded query $\mathbf{q}$ and the mean query point feature $\mathbf{F}_{\alpha}(\mathbf{T}\mathcal{X})$ for a 6-DOF pose $\mathbf{T}$. In practice, we apply min-max normalization to $\mathcal{J}_q$ over all proposals.

4 Results

4.1 Learning to Grasp from Demonstrations

We consider five 6-DOF grasping and placing tasks and provide two demonstrations per task (Fig.4). To label the demonstrations, we load a NeRF-reconstructed point cloud into virtual reality, and use a hand-controller to move a gripper to the desired pose (Fig.2c). We compare the performance of three types of distilled features: DINO ViT, CLIP ViT, and CLIP ResNet. We consider three baselines using NeRF density, intermediate features, and RGB as features, and compare against MIRA [23], a recent work which uses NeRFs to render orthographic viewpoints for pixel-wise affordance predictions. For each task, we evaluate on ten scenes with unseen objects in novel poses, and distractors. We reset the scenes to about the same configuration for each compared method. Please see Appendix A.3 for full details on the experimental setup.

We present the success rates in Table 1 and examples of robot executions in Figure 5. While the baselines achieve respectable performance, they struggle to identify the semantic category of the objects we care about, especially in complex scenes with more distractors. We find that DINO and CLIP feature fields exhibit impressive generalization capabilities and have complementary advantages. DINO has a better understanding of geometry with only $7/19$ failures caused by grasps with inaccurate rotations and occasionally imprecise translations. In contrast, $21/27$ of the failures for CLIP ViT and CLIP ResNet may be attributed to this issue. Given CLIP's training objective, we find that it favors semantic information over geometric features important for grasping and placing. However, DINO struggles with distractor objects which have high feature similarity to the demonstrations, despite not representing the objects and their parts we care about. CLIP struggles far less with distractors due its stronger semantic understanding. To test the limit for our system to work in cluttered environments, we include a scene containing a caterpillar buried under other toys, which would be challenging for off-the-shelf segmentation models (Fig.6b). Our system is able to successfully grasp the caterpillar and drag it out of the pile despite the significant clutter (Fig.6c).

4.2 Language-Guided Object Manipulation

We evaluate the ability to designate objects to manipulate through a variety of language queries using feature fields distilled from CLIP, and reuse the 10 demonstrations across the four object categories in Fig.4. We setup 13 scenes comprising objects from the demonstrated categories, related but out-of-distribution (OOD) categories (e.g., bowls, measuring beakers, utensils), and distractors.

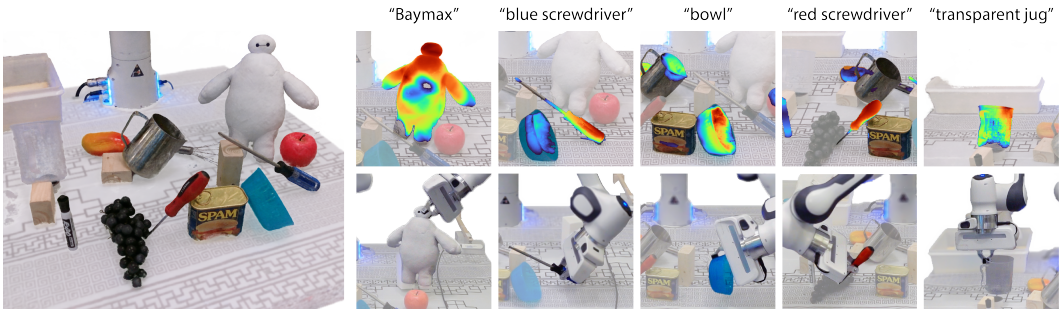


Figure 7: **Language-Guided Manipulation Execution.** (Top Row) Segmentation heatmaps given the language queries. (Bottom Row) Robot executing grasps sequentially without rescanning. CLIP can behave like a bag-of-words, as shown by the bleed to the blue bowl for "blue screwdriver".

	Mug lip	Mug handle	Caterpillar ear	Screwdriver handle	Cup on rack	Total
MIRA [23]	1/10	2/10	6/10	3/10	3/10	15/50
Density	5/10	5/10	**10/10**	2/10	5/10	27/50
Intermediate	2/10	2/10	1/10	3/10	1/10	9/50
RGB	4/10	3/10	9/10	1/10	4/10	21/50
DINO ViT	5/10	4/10	8/10	6/10	**8/10**	31/50
CLIP ViT	7/10	**7/10**	8/10	6/10	6/10	34/50
CLIP ResNet	**9/10**	6/10	9/10	**8/10**	7/10	**39/50**

Table 1: **Success rates on grasping and placing tasks.** We compare the success rates over ten evaluation scenes given two demonstrations for each task. We consider a run successful if the robot grasps or places the correct corresponding object part for the task.

Color	7/10
Material	7/10
Relational	4/10
General	4/10
OOD	9/10
Total	31/50

Table 2: **Success rates of Language-Guided Manipulation.** Language query success rates across semantic categories.

We consider a trial successful if the robot successfully grasps the object specified by the language query. Our robot succeeds in 31 out of 50 language queries, encompassing both general queries (e.g., mug, drying rack) and a diverse range of semantic categories including color, material, and spatial relations (e.g., screwdriver on block). Notably, with only 10 demonstrations, CLIP feature fields exhibit the ability to generalize beyond the demonstrated categories, successfully grasping OOD items including a Baymax plush toy, roll of tape, and scissors. Table 2 breaks down the success rates by category, while Figs. 1 and 7 depict robot executions. See Appendix A.6 for further details. We identify that 6/19 failures stem from poor grasp predictions as encountered in Sec. 4.1, while the remaining 13/19 are due to CLIP features failing or behaving like a bag-of-words and struggling to capture relationships, attributes, and order information within sentences [24]. For queries such as "black screwdriver" and "mug on can of spam", CLIP pays more attention to other black objects and the can of spam, respectively. For similar reasons, retrieving demos via text occasionally requires prompt engineering. One approach to tackle these challenges is ensembling multiple queries and incorporating hard negatives to reject distractors. However, we believe that leveraging the rapid advancements in VLMs holds the most potential for improving performance. Our pipeline enables the integration of new models within minutes without the need to re-collect demonstrations.

Quality and Modeling Time. F3D benefits from the use of patch-level dense features from CLIP, and is able to produce high-quality feature maps without the need to repetitively evaluate the CLIP image model over crops of the image (Fig. 8). We found that patch embeddings from CLIP, similar to those observed in Amir et al. [4], capture local semantics surprisingly well without the use of features from a hierarchy of scales. We compare with LERF [16] in Figure 9.

5 Related Work

Neural descriptor fields generate object-centric neural fields from 3D point cloud data for manipulation [17]. We build upon these approaches, and propose a way to do so with scene-level 3D feature fields that are produced from multi-view RGB images. A number of methods [12, 13, 23]

(a) RGB Image (b) Raw ViT Feat. (c) Distilled Feat.

Figure 8: **Level of Detail.** (a) Mesh strainer and whisk. (b) Raw feature map from DINO ViT, very low in resolution. Colors correspond to PCA of the features. (c) Volumetric Up-Sampling (VUS). Rendering from our feature field has a much higher level of detail. Inset corresponds to the original size of (b) for comparison.

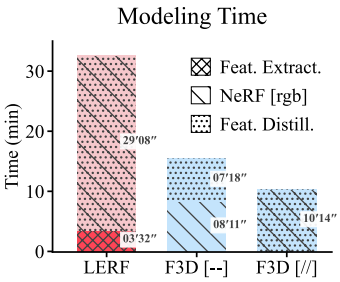


Figure 9: **Timing Benchmark.** Our setup (in blue is more than 3× faster, including time it takes to extract dense embeddings.

use NeRFs to handle objects with thin structure and reflective or transparent materials that are otherwise difficult for traditional structured light sensors. A number of systems produce dense visual descriptors that are useful for manipulation and other robotic applications. Early work in computer vision [25, 26] used dynamic 3D reconstruction from RGB-D videos to provide labels of association between pixels from different video frames. Dense Object Nets [10] eliminate the need for dynamic reconstruction by demonstrating the ability to learn dense descriptors for grasping using multi-view RGB-D images of static scenes [27]. NeRF-Supervision [11] leverages NeRFs for 3D reconstruction, extending descriptor learning to thin structures and reflective materials that pose challenges for depth sensors. In essence, all these methods rely on some form of 3D reconstruction to provide labels for learning dense 2D descriptors, making it difficult to scale to large datasets. Recent results from foundation models [4] show that the DINO ViT, a self-supervised vision transformer (ViT) trained on ImageNet (1 million images), encodes part-level semantics out-of-the-box. These results should not come as a surprise to us, as self-supervised vision models tend to cluster semantically similar parts with each other, even on smaller datasets [10].

There are several other systems that integrate 2D foundation models with 3D, in contexts other than robot manipulation. For example, OpenScene [28] and CLIP-Fields [29] distill 2D features into a neural field by sampling points from existing 3D point clouds or meshes. USA-Net [30] and VLMap [31] both build 2D feature maps from RGB-D images and VLMs for navigation. Our approach uses RGB images and integrates geometric modeling via NeRF with feature fusion in a single pipeline. Distilled feature fields also offer significant compression in space, a gain that becomes more pronounced when modeling high-dimensional features. Feature-realistic neural fusion [32] and ours both require less than 100 MB per scene, a minimal footprint compared to the 50-100s GB required by the surfels used by ConceptFusion [33]. Our work shares many similarities with LERF [16]. The main difference is we develop a method to extract patch-level features from CLIP, which makes our distillation scheme less sensitive to large differences in the physical dimension of objects in the scene.

6 Conclusion

We have illustrated that a way to combine geometric representations with open-ended visual understanding is by distilling 3D feature fields from multiple 2D feature maps from foundation models. When we use distilled feature fields for robot perception and action, they offer out-of-the-box generalization to novel objects, and are robust against variations in appearance and object poses. When the features are sourced from vision-language models, distilled feature fields enable language-guidance with various levels of semantic granularity.

Limitations. We automate taking the RGB images to run over 1 minute and 41 seconds. We did not speed up this part of the process to provide a safety buffer in case of collision. The main rate limit in our system comes from optimizing the NeRF and feature field. Despite many improvements in speed and automation, our system is unable to perceive changes in the scene at real-time.

References

[1] V. Tschernezki, I. Laina, D. Larlus, and A. Vedaldi. Neural Feature Fusion Fields: 3D Distillation of Self-Supervised 2D Image Representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022.

[2] S. Kobayashi, E. Matsumoto, and V. Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[3] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[4] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel. Deep ViT Features as Dense Visual Descriptors. *ECCVW What is Motion For?*, 2022.

[5] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[6] T. Müller, A. Evans, C. Schied, and A. Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics (ToG)*, 2022.

[7] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*, 2022.

[8] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*, 2023.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020.

[10] P. R. Florence, L. Manuelli, and R. Tedrake. Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation. In *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, June 2018.

[11] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola. NeRF-Supervision: Learning dense object descriptors from neural radiance fields. In *IEEE Conference on Robotics and Automation (ICRA)*, 2022.

[12] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg. Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=Bxr45keYrf.

[13] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, and H. Wang. GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[14] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman. Unsupervised Semantic Segmentation by Distilling Feature Correspondences. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SaKO6z6Hl0c.

[15] Y. Rao, W. Zhao, G. Chen, Y. Tang, Z. Zhu, G. Huang, J. Zhou, and J. Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18082–18091, 2022.

[16] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. LERF: Language Embedded Radiance Fields. *arXiv preprint arXiv:2303.09553*, 2023.

[17] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural Descriptor Fields: SE(3)-Equivariant Object Representations for Manipulation. In *ICRA*, 2022.

[18] A. Simeonov, Y. Du, L. Yen-Chen, , A. Rodriguez, , L. P. Kaelbling, T. L. Perez, and P. Agrawal. SE(3)-Equivariant Relational Rearrangement with Neural Descriptor Fields. In *Conference on Robot Learning (CoRL)*. PMLR, 2022.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016.

[21] C. Garrett. PyBullet Planning. https://github.com/caelan/pybullet-planning, 2018.

[22] N. Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[23] L. Yen-Chen, P. Florence, A. Zeng, J. T. Barron, Y. Du, W.-C. Ma, A. Simeonov, A. R. Garcia, and P. Isola. MIRA: Mental imagery for robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.

[24] M. Yuksekgonul, F. Bianchi, P. Kalluri, D. Jurafsky, and J. Zou. When and Why Vision-Language Models Behave like Bags-Of-Words, and What to Do About It? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KRLUvxh8uaX.

[25] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal Correspondence Network. In *Advances in Neural Information Processing Systems*, June 2016.

[26] T. Schmidt, R. Newcombe, and D. Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robot. Autom. Lett.*, Apr. 2017.

[27] P. Florence, L. Manuelli, and R. Tedrake. Self-Supervised Correspondence in Visuomotor Policy Learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, Apr. 2020.

[28] S. Peng, K. Genova, C. m. Jiang, A. Tagliasacchi, M. Pollefeys, and T. Funkhouser. OpenScene: 3D Scene Understanding with Open Vocabularies. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nov. 2022.

[29] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv: Arxiv-2210.05663*, 2022.

[30] B. Bolte, A. S. Wang, J. Yang, M. Mukadam, M. Kalakrishnan, and C. Paxton. USA-Net: Unified Semantic and Affordance Representations for Robot Memory. *ArXiv*, abs/2304.12164, 2023.

[31] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual Language Maps for Robot Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

[32] K. Mazur, E. Sucar, and A. Davison. Feature-Realistic Neural Fusion for Real-Time, Open Set Scene Understanding. In *2023 International Conference on Robotics and Automation (ICRA)*, 2023.

[33] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, J. Tenenbaum, C. de Melo, M. Krishna, L. Paull, F. Shkurti, and A. Torralba. Conceptfusion: Open-set multimodal 3d mapping. *arXiv*, 2023.

[34] N. Max. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 1(2): 99–108, June 1995.

[35] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[36] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[38] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[39] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. NeRF−−: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064*, 2021.