

Splat-MOVER: Multi-Stage, Open-Vocabulary Robotic Manipulation via Editable Gaussian Splatting

Anonymous Author(s)

Affiliation

Address

email

1 Appendix A Affordance-and-Semantic-Knowledge Gaussian Splatting

2 Here, we provide additional details of the distilled knowledge in ASK-Splat.

3 A.1 Grounding Language Semantics in 3D Gaussian Splatting

4 In this work, we utilize feedforward MLPs in defining the encoder and decoder with $l = 3 \ll C$.
5 However, we note that larger values of l generally result in more expressive semantic scene representations, at the expense of increased memory and rendering costs. We train the autoencoder with the
6 loss function \mathcal{L}_g , given by:
7

$$\mathcal{L}_g = \kappa_g \sum_{i=1}^{|\mathcal{D}|} \|g_\theta^{\text{dec}}(g_\phi^{\text{enc}}(f_{\text{gt},i})) - f_{\text{gt},i}\|_2^2 + \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} (1 - \psi(g_\theta^{\text{dec}}(g_\phi^{\text{enc}}(f_{\text{gt},i})), f_{\text{gt},i})), \quad (1)$$

8 where $g_\theta^{\text{dec}}(g_\phi^{\text{enc}}(\cdot))$ represents the composition of the encoder and decoder that outputs the reconstruction of its inputs, $\psi : \mathbb{R}^{U \times V \times C} \times \mathbb{R}^{U \times V \times C} \mapsto \mathbb{R}$ denotes the cosine-similarity function (where
9 we note that ψ applies to inputs of arbitrary height and width), and \mathcal{D} denotes the dataset of images
10 used in training the Gaussian Splatting scene, with $f_{\text{gt},i}$ denoting the ground-truth semantic features
11 of image i . The first term in (1) represents the mean-squared-error (MSE) reconstruction loss with
12 $\kappa_g \in \mathbb{R}_{++}$ denoting the constant associated with this term, while the second term represents the
13 cosine-similarity loss between the ground-truth embeddings and the reconstruction.
14

15 Given a trained encoder, we map the ground-truth image embeddings from CLIP to the lower-dimensional latent space and distill the lower-dimensional embeddings into the Gaussian Splatting
16 representation. We assign a semantic feature $f_s \in \mathbb{R}^l$ to each Gaussian. To render 2D semantic
17 feature maps of the scene, we utilize the same tile-based rasterization procedure presented in [1],
18 culling 3D Gaussians whose 99% confidence ellipsoid do not intersect the view frustum associated
19 with the pose of the camera. We optimize the semantic feature parameters using the loss function:
20

$$\mathcal{L}_s = \kappa_s \sum_{i=1}^{|\mathcal{D}|} \|\hat{\mathcal{I}}_i^s - g_\phi^{\text{enc}}(f_{\text{gt},i})\|_2^2 + \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} (1 - \psi(\hat{\mathcal{I}}_i^s - g_\phi^{\text{enc}}(f_{\text{gt},i}))), \quad (2)$$

21 where $\hat{\mathcal{I}}_i^s \in \mathbb{R}^{H \times W \times l}$ denotes the 2D semantic feature map rendered from the Gaussian Splats and
22 $\kappa_s \in \mathbb{R}_{++}$ denotes the constant term in the MSE loss, given by the first component of \mathcal{L}_s . Although
23 not explicitly stated in (2), we resize the output of g_ϕ^{enc} using bilinear interpolation to obtain a
24 ground-truth semantic map of compatible dimensions.

25 Given a good initialization of the Gaussians (e.g., when the sparse point cloud from structure-from-motion is utilized in initializing the Gaussians), the semantic feature parameters associated with each
26 Gaussian can be trained simultaneously with other spatial and visual-related parameters associated
27 with the Gaussians, along with the autoencoder’s parameters. Nonetheless, empirical evaluations
28 suggest that a sequential training procedure in which the semantic parameters are trained after the
29

non-semantic parameters of the Gaussians have been trained yields better localized semantic feature maps. We hypothesize that this observation may result from more consistent grounding of the semantic features.

Now, we present our approach to generating semantic feature maps of the scene given a natural-language query. We compute the text embedding of the query using CLIP, which we use in evaluating the similarity between the specified query and the objects in scene. We utilize the cosine-similarity metric, which is widely used in prior work [2, 3]. Consistent with prior work, we allow for the specification of *negative* queries to help distinguish between dissimilar objects and the object of interest described by a *positive* query. However, we note that, in practice, a positive query suffices without negative queries. We compute the embeddings of each item in the set of negative queries and positive queries, and subsequently compute the cosine similarity between the predicted semantic feature given by the Gaussian Splats and each query in the set of negative and positive queries. Finally, the similarity score between a feature point p and the positive query is given by:

$$\text{sim}(\mathcal{Q}_s, \mathcal{Q}'_s) = \min_{i \in |\mathcal{Q}'_s|} \gamma(p, \nu_a(\mathcal{Q}_s), \nu_b(q'_{s,i})), \quad (3)$$

where \mathcal{Q} denotes the set of positive queries, $\mathcal{Q}'_s = \{q'_{s,i}, \forall i \in [|\mathcal{Q}'_s|]\}$ denotes the set of negative queries, $\nu_a : \mathcal{S} \mapsto \mathbb{R}$ computes the average semantic embedding of a set of text prompts $\bar{s} \in \mathcal{S}$, $\nu_b(q'_{s,i})$ represents the semantic embedding of the negative query $q'_{s,i}$, and $\gamma : \mathbb{R}^3 \times \mathbb{R}^C \times \mathbb{R}^C \mapsto \mathbb{R}$ represents the pairwise softmax function over the positive query embedding and the i th negative query embedding at the 3D feature point, outputting the probability associated with the positive query embedding. In general, when rendering the semantic similarity maps, we apply a threshold of 0.5 to the similarity score computed in (3) to distinguish feature points associated with the query from dissimilar ones.

A.2 Grounding Affordance in 3D Gaussian Splatting

We embed grasp affordances in ASK-Splat. During training, we utilize the same tile-based rasterization procedure discussed in Section A.1 in rendering the 2D visual grasp affordance of the scene and optimize the grasp affordance parameter β using the loss function: $\mathcal{L}_\beta = \kappa_\beta \sum_{i=1}^{|\mathcal{D}|} \left\| \hat{\mathcal{I}}_i^\beta - \mathcal{I}_i^\beta \right\|_2^2$, which represents the MSE loss between the ground-truth 2D visual grasp affordance $\mathcal{I}_i^\beta \in \mathbb{R}^{H \times W \times 1}$ and the rendered visual grasp affordance $\hat{\mathcal{I}}_i^\beta \in \mathbb{R}^{H \times W \times 1}$. We optimize the affordance parameters concurrently with the non-semantic parameters of each Gaussian. From a trained ASK-Splat scene, we can generate dense 2D visual grasp affordance maps, as well as sparser 3D visual grasp affordance maps, by directly evaluating the affordance score associated with each Gaussian.

Appendix B Scene-Editing-Enabled Gaussian Splatting

We present the components that make up SEE-Splat, our module for Scene-Editing-Enabled Gaussian Splatting representations, that enables the identification and localization of relevant objects within a scene for insertion, removal, or modification of the object’s visual or spatial properties.

B.1 Semantic Localization via ASK-Splat

Given a natural-language query specifying an object of interest, SEE-Splat leverages ASK-Splat to identify semantically relevant Gaussians, as discussed in Section A.1. In the main paper (cf. Fig. 3), we show the localization of an electric stove, saucepan, and a fruit in a real-world Cooking scene. To improve the localization accuracy, the text prompt can include the geometric and visual properties of the object, such as its color, in addition to its semantic class. At this stage, SEE-Splat generates a semantic similarity map, from which relevant Gaussians are extracted, given a threshold on the semantic score.

72 B.2 Masking the Gaussians in SEE-Splat

73 Given a semantic similarity map of the scene, SEE-Splat generates a mask identifying the Gaussians
74 relevant to the specified object. This procedure begins with thresholding the semantic scores of each
75 Gaussian to remove dissimilar Gaussians from the set of relevant Gaussians, which creates a sparse
76 point cloud of the relevant Gaussians, constructed from the means of these Gaussians. However,
77 photo-realistic rendering of Gaussian environments require denser point clouds. Consequently, SEE-
78 Splat lifts the features of the point cloud from the 3D Euclidean space to a 7D feature space, by
79 augmenting each point in the point cloud with its RGB color and semantic score. Subsequently,
80 SEE-Splat identifies all neighboring points in the scene within a specified distance of the point cloud
81 in the 7D feature space using an efficient KD-tree query. SEE-Splat incorporates these points into
82 the point cloud to create a denser point cloud, comprising of all semantically-relevant Gaussians,
83 while removing outliers from the set of points. In the main paper (cf. Fig. 3), we show the Gaussians
84 extracted by SEE-Splat, as a point cloud with well-defined geometry, given a natural-language query
85 for each object.

86 B.3 Editing the Gaussians in SEE-Splat

87 Leveraging the Gaussian primitives in ASK-Splat, SEE-Splat enables real-time scene-editing by
88 inserting new Gaussians into the scene, removing Gaussians, and modifying the properties of the
89 Gaussians, to reflect (or simulate) changes in the real-world. SEE-Splat supports seamless insertion
90 and removal of Gaussians by introducing or deleting the relevant Gaussians from the set of Gaussians
91 representing the scene, respectively. In addition, SEE-Splat supports both rigid and non-rigid
92 transformation of the Gaussians, enabling simulated motion of the Gaussians, as well as changes to
93 the shape of the Gaussians via non-isometric scaling. Specifically, given a function specifying the
94 transformation $\xi : \mathcal{G}_s \mapsto \mathcal{G}_s$ (where \mathcal{G}_s represents the space of the Gaussian primitives), SEE-Splat
95 updates the spatial attributes of the relevant Gaussians by applying ξ to these Gaussians. In the case of
96 rigid transformations, ξ can be described by an SE(3) transformation matrix, specifying rotation and
97 translation of the Gaussians. We can render the edited scene to generate photo-realistic visualizations.
98 Although, we do not consider physics-based simulations in this work, we note that physics can be
99 incorporated into SEE-Splat to achieve realism. We expound on this point in or discussion on the
100 limitations of SEE-Splat.

101 Deletion and transformation of the Gaussians introduces artifacts into the scene representation,
102 degrading its photo-realistic qualities. To address this challenge, SEE-Splat enables 3D Gaussian
103 infilling by introducing new Gaussians with similar attributes in regions with missing geometry,
104 which we illustrate in Appendix A. Figure 1 provides an illustration of such artifacts (e.g., the hole in
105 the table), when the scene is edited to visualize the effects of moving the saucepan to the electric
106 stove. Through 3D Gaussian infilling, SEE-Splat generates a photorealistic rendering of the edited
107 scene, eliminating these artifacts.

108 Appendix C Grasping and Manipulation with Splat-MOVER

109 We present Grasp-Splat and discuss its application to multi-stage robotic manipulation via Splat-
110 MOVER.

111 C.1 Grasp-Splat for Grasp Proposal

112 We note that the grasps generated by GraspNet are not always ideal. For example, the grasps generated
113 by GraspNet in Figure 2 are either infeasible or challenging to execute. As a result, Grasp-Splat ranks
114 the grasps proposed by GraspNet based on the grasp scores obtained from ASK-Splat. By leveraging
115 the affordance score associated with each grasp pose, Grasp-Splat identifies grasp configurations that
116 are more likely to succeed, depicted in Figure 2.

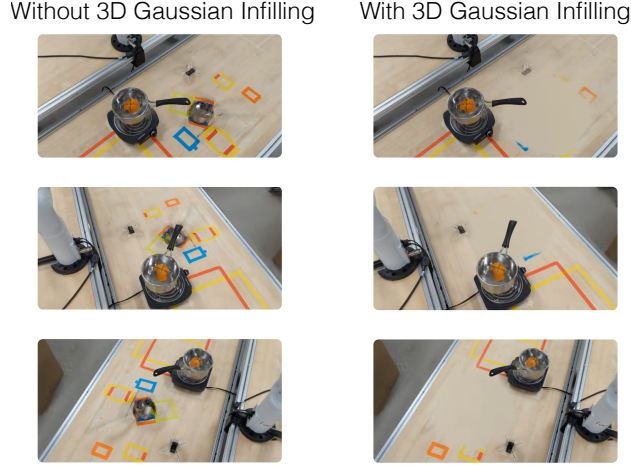


Figure 1: 3D Gaussian Infilling in SEE-Splat: (Left) In general, without 3D Gaussian infilling, transformation of the Gaussians (e.g., moving the saucepan from the table to the electric stove) introduces artifacts, such as the hole in the table after moving the saucepan. (Right) Via 3D Gaussian infilling, SEE-Splat generates photorealistic renderings of the edited scene.

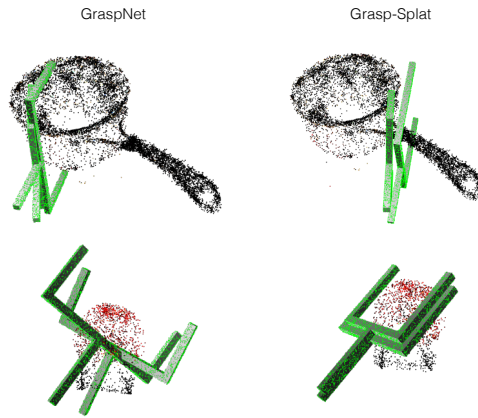


Figure 2: The top-two candidate grasps proposed by (left) GraspNet and (right) Grasp-Splat, leveraging the grasp affordance of each object. Qualitatively, the grasps generated by Grasp-Splat are more likely to succeed, compared to the grasps generated solely from Grasp-Net. Further, the grasps proposed by Grasp-Splat are better localized on the handle of the saucepan.

117 C.2 Multi-Stage Robotic Manipulation

118 For multi-stage robotic manipulation, we begin by decomposing the manipulation task into stages.
 119 Our approach supports the specification of the stages comprising the task by a human or by a large
 120 language model (LLM). In the case where the natural-language description of the task does not
 121 specify the stages involved in the task, we query an LLM for the stages required to complete the
 122 manipulation task. For each stage in the manipulation task, we utilize ASK-Splat, SEE-Splat, and
 123 Grasp-Splat to identify the relevant object and generate candidate grasp poses. Likewise, we query
 124 ASK-Splat for the target location for placing the object. We evaluate the feasibility of each candidate
 125 grasp using an off-the-shelf motion planner for the robotic manipulator, inputting the point cloud
 126 of the scene, extracted from ASK-Splat and SEE-Splat, into the motion planner, which the motion
 127 planner uses for collision detection during motion planning. We execute the top candidate grasps,
 128 moving on to the next if the robot motion planner fails to compute a solution to execute the selected
 129 grasp.

130 We execute the motion plan returned by the motion planner on the robotic manipulator. We note that
 131 the end-effector trajectory can be published to SEE-Splat for real-time visualization of the task in the
 132 virtual scene. In this case, we can apply the relative transformation between consecutive end-effector
 133 poses to the spatial attributes of the Gaussian associated with the object being manipulated. In
 134 addition, we note that alternative approaches exist for computing the relative transformations of the
 135 object between consecutive frames. For example, if object-tracking information is available from
 136 sensors in the scene, SEE-Splat could leverage this information to update the spatial attributes of
 137 the Gaussians, rendering a video showing the real-time changes in the scene of the manipulator,
 138 including the motion of the object, as the manipulation task progresses. We proceed to the next stage
 139 in the manipulation task at the conclusion of the current stage, repeating the same procedures with
 140 the updated representation of the scene provided by SEE-Splat.

141 **Appendix D Evaluations**

142 We present additional experimental results of ASK-Splat, SEE-Splat, and Splat-MOVER in open-
 143 vocabulary, multi-stage robotic manipulation problems, including a discussion of the experimental
 144 setup.

145 **D.1 Experimental Setup**

146 **D.1.1 ASK-Splat**

147 We distill grasp affordances from the vision-affordance foundation model VRB [4], which is trained
 148 on the EPIC-KITCHENS dataset [5], consisting of videos of humans performing kitchen tasks, such
 149 as cutting fruits and vegetables. We note that the generalization of the affordance knowledge in
 150 ASK-Splat is limited by that of VRB, the underlying foundation model. VRB utilizes Language
 151 Segment-Anything (LangSAM) [6], which requires the specification of objects within each image for
 152 which it predicts the contact locations and motion direction after contact. This requirement is not
 153 limiting, in practice, as end-to-end object detectors that provide bounding boxes for all objects in the
 154 scene [7, 8, 9] could be used. We distill the grasp affordance scores from the heatmaps computed by
 155 VRB and the semantic embeddings from the vision-language foundation model RN50 \times 64, CLIP-
 156 ResNet model [10]. We implement ASK-Splat in Nerfstudio [11]. To train ASK-Splat, we record a
 157 video of each scene using a smartphone and utilize the training API available in Nerfstudio, using the
 158 sparse point cloud computed via structure-from-motion [12] for initialization.

159 **D.1.2 Scenes**

160 We consider only real-world scenes in our experiments, including a *Kitchen* scene (consisting
 161 of common kitchen cookware such as saucepans, chopping boards, and knives); *Cleaning* scene
 162 (consisting of common household cleaners such as disinfectant wipes, dish soaps, and cleaning
 163 sprays); *Meal* scene (consisting of cutlery such as plates, spoons, forks, and cups); *Random* scene
 164 (consisting of random items such as a pair of scissors, chess pieces, and keyholders); and a *Workshop*
 165 scene (consisting of tools such as a power drill, work mat, and scraper). Figure 3 shows these scenes.
 166 We note that the *Workshop* and *Random* scenes contain out-of-distribution objects with respect to the
 167 EPIC-KITCHENS dataset (i.e., objects not found in a typical kitchen), such as the power drill and the
 168 chess pieces.

169 **D.1.3 Splat-MOVER**

170 We consider the multi-stage robotic manipulation task where the robot must sequentially pick and
 171 place two different objects and move them to a common goal location. The task is specified by a user
 172 that provides an open-vocabulary command, e.g., “Pick up the saucepan and move it to the burner,
 173 then pick up the lid and put it on the saucepan.” For simplicity, we limit the task to two sequential
 174 pick-and-place maneuvers. However, we note that Splat-MOVER does not impose this limitation
 175 and is amenable to longer multi-stage manipulation tasks. Furthermore, we consider three adjacency

goal location primitives (“on”, “next to”, and “inside”) for the second object where each primitive is defined based on the geometry of the first object.

Specifically, we evaluate Splat-MOVER in four multi-stage manipulation tasks across three scenes: the *Kitchen*, *Cleaning*, and *Workshop* scenes. In the *Kitchen* scene, we consider a *Cooking* task where the robot is asked to place a saucepan on an electric burner (Stage 1) and subsequently place a fruit inside the saucepan (Stage 2). Further, in the *Kitchen* scene, we consider a *Chopping* task where the robot is asked to place a knife on a chopping board (Stage 1) and subsequently place a fruit next to the knife (Stage 2). We consider a *Cleaning* task (in the *Cleaning* scene), where the robot is asked to place a cleaning spray in a bin (Stage 1) and subsequently place a sponge next to the cleaning spray (Stage 2). Lastly, in the *Workshop* scene, the robot is asked to place a power drill on a work mat (Stage 1) and subsequently place a wooden block next to the drill (Stage 2), which we refer to as the *Workshop* task.

D.1.4 Hardware Experiments

We implement Splat-MOVER in grasping and placing tasks on a Kinova Gen3 robot, equipped with a Robotiq parallel-jaw gripper. The Kinova robot is a 7-DoF robot with a maximum reach of 902 mm. We interface with the robot using the Robot Operating System (ROS), through which we send waypoints, which are tracked by the default low-level controllers provided by the robot. We utilize the MoveIt ROS package [13] for motion planning for the Kinova robot given a specified grasp pose. At each stage of the manipulation task, we extract a point cloud and a mesh from ASK-Splat and SEE-Splat, reflecting the progress in the task up to that stage, which we use as the environment representation within MoveIt for collision avoidance during planning.

D.2 ASK-Splat Representation

We train ASK-Splat on a number of different environments and evaluate the grasp affordance and semantic segmentation of the resulting Gaussian Splats. In Figure 3, we show the RGB image, grasp affordance heatmap computed by VRB, and the grasp affordance heatmap rendered from ASK-Splat composited with the rendered RGB image. The heatmap shows the regions in each object amenable to grasping. Qualitatively, from Figure 3, ASK-Splat encodes the grasp affordance given by VRB, identifying reasonable regions on each object for grasping. Although VRB provides the 2D motion direction associated with each grasp affordance region, we do not distill this knowledge into ASK-Splat, as we found the 2D motion directions to be quite noisy and relatively uninformative.

We compute the Structural Similarity Index (SSIM) for each scene to assess the quality of the distilled affordance compared to the VRB-generated grasp affordance. The SSIM metric ranges between -1 (indicating greater dissimilarity) to 1 (indicating greater similarity). As expected, the Workshop scene yields the smallest SSIM value of $0.592 \pm 7.20e^{-2}$, recalling that the objects in this scene, such as the power drill and the scraper, are outside the training distribution of the VRB model. Nevertheless, the model shows relatively-good generalization performance, given that the grasp affordance region lies around the handle of the drill, shown in Figure 3 (bottom row). Likewise, the Meal scene achieves the highest SSIM score of $0.681 \pm 8.91e^{-2}$, noting that the objects in the scene can be found in the dataset used in training the VRB model. Further, the Cleaning, Kitchen, and Random scenes achieve SSIM scores of $0.648 \pm 9.06e^{-2}$, $0.647 \pm 1.30e^{-1}$, and $0.614 \pm 8.37e^{-2}$, respectively.

Figure 4 shows the semantic masks generated by ASK-Splat across different scenes. Given a natural-language query, ASK-Splat localizes the relevant object in the scene based on the cosine-similarity of the Gaussians to the query. In Figure 4, ASK-Splat identifies the *salt shaker*, *flower*, and pair of *scissors*. However, the success of robotic manipulation tasks depend on the integration of semantic scene understanding with grasp affordance. As such, we show the semantic-affordance masks generated by ASK-Splat in Figure 4. With the semantic-affordance masks, a robot not only has the ability to identify a relevant object to grasp, the robot can also identify where on the relevant object to grasp.



Figure 3: Grasp affordance for a *Kitchen* scene, *Cleaning* scene, *Meal* scene, *Random* scene, and *Workshop* scene (from top-to-bottom). We show the RGB image, grasp affordance as predicted by the vision-affordance foundation model (VRB), and the grasp affordance from ASK-Splat from novel views (from left-to-right).

D.3 Splat-MOVER for Multi-Stage Robotic Manipulation

We compare Splat-MOVER to prior work LERF-TOGO [10] and F3RM [14] in four tasks: the *Cooking* task, *Chopping* task, *Cleaning* task, and *Workshop* task, described in Section D.1.3. Figure 5 shows a few candidate grasps proposed by GraspNet, F3RM, LERF-TOGO, and Grasp-Splat for each of these objects. GraspNet does not consider the semantic features of the object in generating candidate grasps; as a result, the proposed grasps are not localized in regions where a human might grasp the object, unlike the candidate grasps proposed by F3RM, LERF-TOGO, and Grasp-Splat, which generate grasps closer to the handle of the respective objects. For example, the proposed grasps lie relatively close to the handle of the saucepan and the knife. F3RM and LERF-TOGO generate candidate grasp conditioned on a text prompt identifying the region to grasp the object (such as its handle) provided by a human operator or an LLM (in LERF-TOGO) or from a dataset of human demonstrations (in F3RM). In contrast, Grasp-Splat does not require any external guidance to generate candidate grasps of similar quality, harnessing the grasp affordances provided by ASK-Splat. We summarize the capabilities of each of these methods in Table 1.

Table 1: Representation Capabilities of LERF-TOGO [10], F3RM [14], and Splat-MOVER

Capabilities	Semantic Knowledge	Affordance Knowledge	Scene-Editing
LERF-TOGO [10]	✓	✗	✗
F3RM [14]	✓	✗	✗
Splat-MOVER (ours)	✓	✓	✓

In addition, we evaluate the pick-and-place success rate of all the methods in the *Cooking* task, *Chopping* task, *Cleaning* task, and *Workshop* task, where the place success rate is conditioned on the number of successful trials in picking the object. Table 2 provides the pick-and-place success rates in the Chopping task. Splat-MOVER achieves the highest pick success rate (85%) in Stage 1 of the task. Although F3RM achieves the highest place success rate, it achieves a much lower pick success rate compared to Splat-MOVER. In addition, in the Cleaning and Workshop tasks, Splat-MOVER achieves the highest success rates in the first stage of each task, and further achieves relatively high

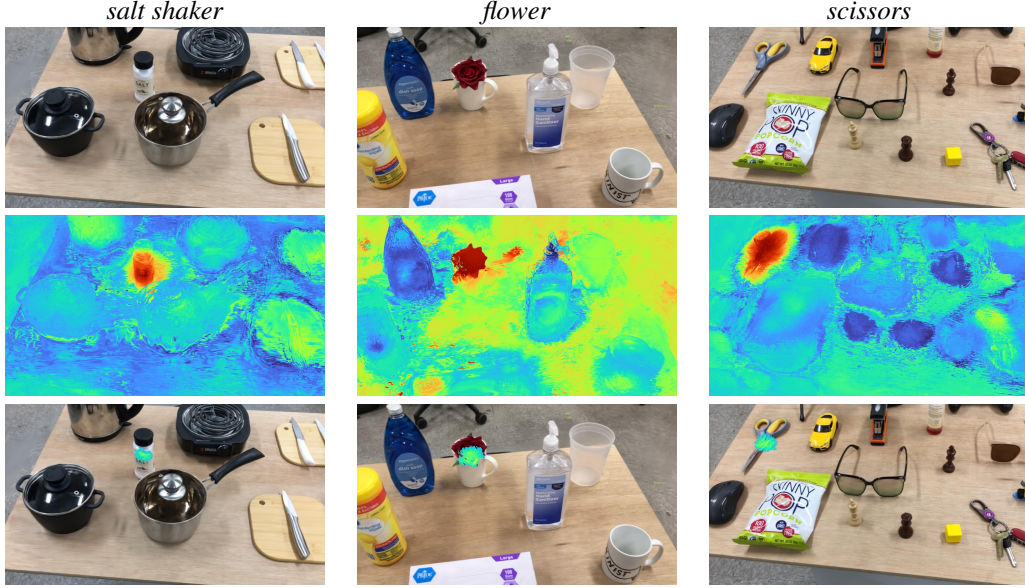


Figure 4: Affordance and Language Semantics in ASK-Splat: Given natural-language queries, ASK-Splat renders: (top-row) RGB images, (middle-row) semantic masks of the scene, and (bottom-row) localized grasp affordance regions, for example, for a *salt shaker* in the *kitchen* scene, a *flower* in the *cleaning* scene, and a pair of *scissors* in the *random* scene, evaluated at novel views in ASK-Splat. The natural-language query for each object is noted in italics.

success rates in the second stage of each task, shown in Tables 3 and 4. LERF-TOGO achieves a perfect success rate in picking up the power drill in the first stage of the Workshop task. Since LERF-TOGO and F3RM are not amenable to multi-stage manipulation tasks, we cannot evaluate the success rate of these methods for the entire manipulation task. In contrast, Splat-MOVER enables multi-stage robotic manipulation, achieving a task success rate of 40%, 65%, 70%, and 80% in the Cooking, Chopping, Cleaning, and Workshop tasks, respectively. We note that the Cooking task is the most challenging task, compared to the other tasks, given the little margin of error tolerated in placing the saucepan on the electric burner.

Table 2: Pick and Place Success Rates in a two-stage manipulation Chopping task, where the robot must move a knife to a chopping board (Stage 1), then move a fruit next to the knife (Stage 2).

Methods	Stage 1		Stage 2	
	Pick Success (%)	Place Success (%)	Pick Success (%)	Place Success (%)
LERF-TOGO [10]	35	N/A	N/A	N/A
F3RM [14]	60	91.67	N/A	N/A
Splat-MOVER (ours)	85	82.35	65	100

Table 3: Pick and Place Success Rates in a two-stage manipulation Cleaning task, where the robot must move a cleaning spray into a bin (Stage 1), then move a sponge next to the cleaning spray inside the bin (Stage 2).

Methods	Stage 1		Stage 2	
	Pick Success (%)	Place Success (%)	Pick Success (%)	Place Success (%)
LERF-TOGO [10]	25	N/A	N/A	N/A
F3RM [14]	75	6.67	N/A	N/A
Splat-MOVER (ours)	90	83.33	70	100

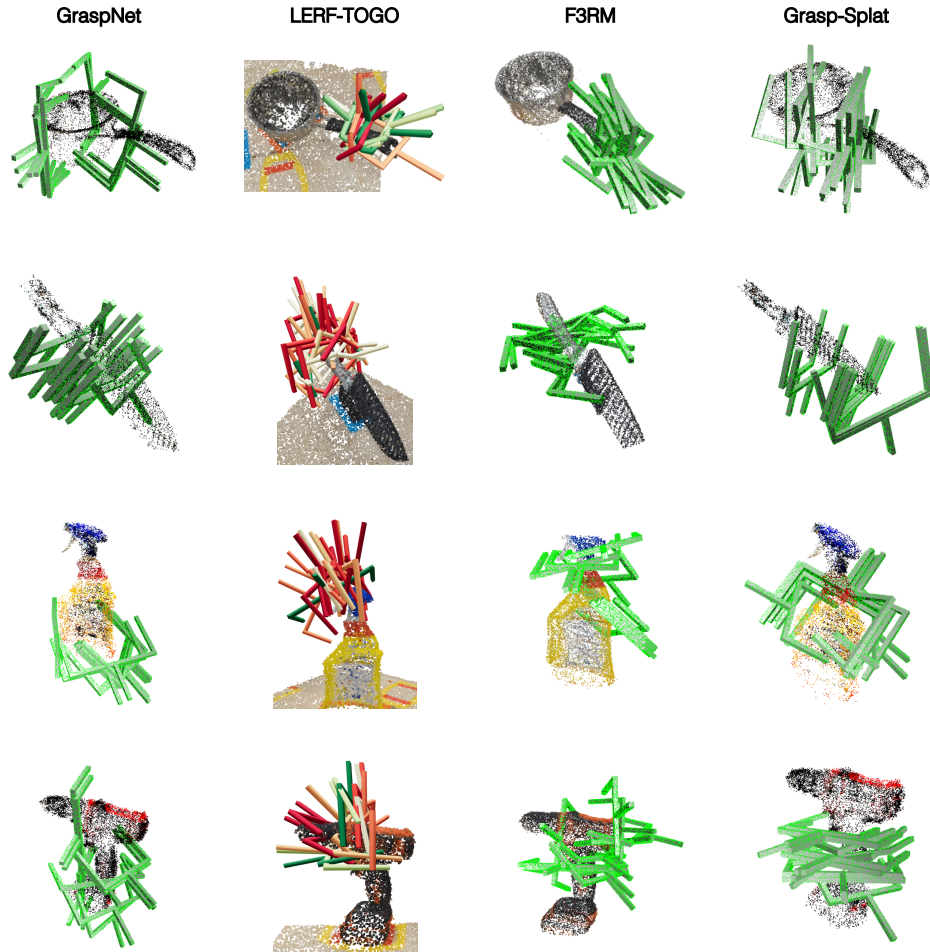


Figure 5: Candidate grasps for a *saucepan*, *knife in a guard*, *cleaning spray*, and *power drill* (from top-to-bottom), generated by GraspNet, LERF-TOGO, F3RM, and Grasp-Splat (from left-to-right). Although LERF-TOGO and F3RM require the specification of a grasp location from an operator, an LLM, or via human demonstrations to generate more-promising candidate grasps, Grasp-Splat generates candidate grasps of similar or better quality without requiring external guidance.

Table 4: Pick and Place Success Rates in a two-stage manipulation in a Workshop task, where the robot is tasked with moving a power drill onto a work mat (Stage 1), followed by moving a wooden block next to the drill on the work mat (Stage 2).

Methods	Stage 1		Stage 2	
	Pick Success (%)	Place Success (%)	Pick Success (%)	Place Success (%)
LERF-TOGO [10]	100	N/A	N/A	N/A
F3RM [14]	70	7.14	N/A	N/A
Splat-MOVER (ours)	95	94.74	85	88.24

References

- [1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.
- [2] 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 (ICML)*, pages 8748–8763. PMLR, 2021.
- [3] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. LERF: Language embedded radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19729–19739, 2023.
- [4] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13778–13790, 2023.
- [5] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European conference on computer vision (ECCV)*, pages 720–736, 2018.
- [6] L. Medeiros. Language Segment-Anything. <https://github.com/luca-medeiros/lang-segment-anything>, 2023.
- [7] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7373–7382, 2021.
- [8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.
- [9] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.
- [10] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *7th Annual Conference on Robot Learning (CoRL)*, pages 178–200. PMLR, 2023.
- [11] M. Tancik, E. Weber, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, A. Kanazawa, and E. Ng. Nerfstudio: A framework for neural radiance field development. In *SIGGRAPH*, 2023.
- [12] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [13] D. Coleman, I. A. Şucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of Software Engineering for Robotics*, 5(1): 3–16, 2014.
- [14] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola. Distilled feature fields enable few-shot language-guided manipulation. In *7th Annual Conference on Robot Learning (CoRL)*, 2023.