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

Ola Shorinwa[1⋆], Johnathan Tucker[1⋆], Aliyah Smith[1], Aiden Swann[2], Timothy Chen[1], Roya Firoozi[1], Monroe Kennedy III[2], Mac Schwager[1]

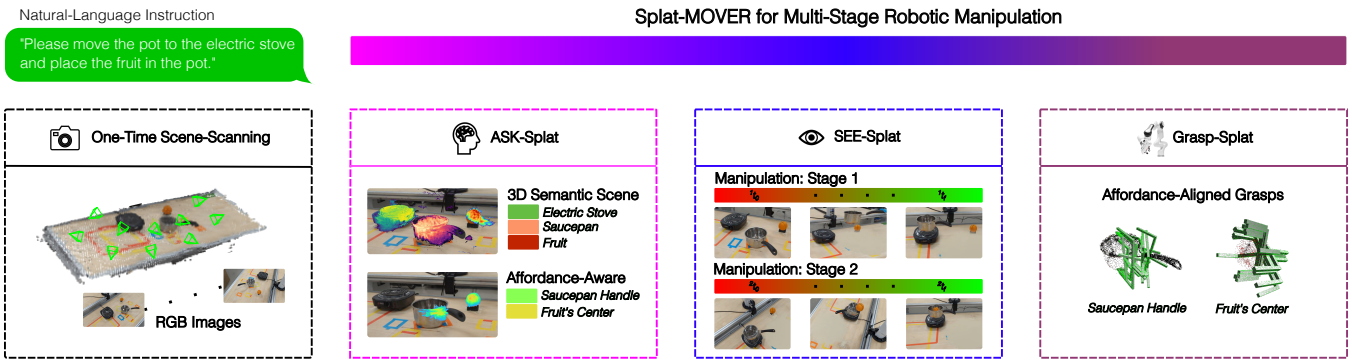

Fig. 1: Splat-MOVER enables language-guided, multi-stage robotic manipulation, through an affordance-and-semantic-aware scene representation ASK-Splat, a real-time scene-editing module SEE-Splat, and a grasp-generation module Grasp-Splat.

***Abstract*— We present Splat-MOVER, a modular robotics stack for open-vocabulary robotic manipulation, which leverages the editability of Gaussian Splatting (GSplat) scene representations to enable multi-stage manipulation tasks. Splat-MOVER consists of: (i) *ASK-Splat*, a GSplat representation that distills latent codes for language semantics and grasp affordance into the 3D scene. ASK-Splat enables geometric, semantic, and affordance understanding of 3D scenes, which is critical for many robotics tasks; (ii) *SEE-Splat*, a real-time scene-editing module using 3D semantic masking and infilling to visualize the motions of objects that result from robot interactions in the real-world. SEE-Splat creates a "digital twin" of the evolving environment throughout the manipulation task; and (iii) *Grasp-Splat*, a grasp generation module that uses ASK-Splat and SEE-Splat to propose candidate grasps for open-world objects. ASK-Splat is trained in real-time from RGB images in a brief scanning phase prior to operation, while SEE-Splat and Grasp-Splat run in real-time during operation. We demonstrate the superior performance of Splat-MOVER in hardware experiments on a Kinova robot compared to two recent baselines in four single-stage, open-vocabulary manipulation tasks, as well as in four multi-stage manipulation tasks using the edited scene to reflect scene changes due to prior manipulation stages, which is not possible with the existing baselines. Code for this project and a link to the project page will be made available soon.**

***Index Terms*— Gaussian Splatting, Robotic Grasping, Robotic Manipulation, Scene Editing**

⋆ The co-first authors contributed equally.

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA {jatucker,shorinwa,chengine,rfiroozi,aliyah1,schwager}@stanford.edu

[2]Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA {swann, monroek}@stanford.edu

This work was supported in part by DARPA grant HR001120C0107, NSF Graduate Research Fellowship DGE-1656518 and DGE-2146755, NSF Grant 2220867, and by a gift from Meta. We are grateful for this support.

Toyota Research Institute provided funds to support this work.

I. Introduction

Robotic manipulation requires spatial and semantic understanding of the scene in which a robot is operating. In particular, a robot must be able to identify the location and geometry of objects in its environment based on their semantic attributes. However, spatial and semantic scene understanding alone is often inadequate in robotic manipulation. In many cases, it is also critical to know the best place to grasp the object to perform the required task, that is, to be able to detect and localize grasp affordances on the object [1], [2]. Furthermore, for multi-stage manipulation tasks, where the pre-conditions for the next stage are met through the success of a previous stage, it is also essential to continually update the 3D scene model to reflect the objects' motions and rearrangements due to the robot's previous actions, to have an accurate scene representation that is useful for subsequent task stages.

Consequently, in this work, we introduce Splat-MOVER, a modular robotics stack for **M**ulti-Stage, **O**pen-**V**ocabulary **R**obotic Manipulation via **E**ditable Gaussian Splatting. Splat-MOVER consists of three modules: (i) a 3D **A**ffordance-and-**S**emantic **K**nowledge Gaussian Splatting scene representation (ASK-Splat), (ii) a **S**cene-**E**diting-**E**nabled module for Gaussian Splatting scenes (SEE-Splat), and (iii) a Grasp-Generation module (Grasp-Splat) that uses ASK-Splat and SEE-Splat to plan grasps in multi-stage manipulation tasks.

ASK-Splat encodes semantic and affordance knowledge distilled from 2D foundation models in the 3D GSplat scene representation. For semantic knowledge, we distill vision-language features from CLIP [3], a 2D foundation model for predicting image-text similarity. We leverage the

feature distillation method in [4], [5] to extract pixel-wise image features from CLIP and embed them as attributes of the Gaussians during training of the GSplat representation. Further, we distill grasp affordances from a pre-trained 2D vision-based affordance model, Vision-Robotics Bridge (VRB) [6], which predicts a set of contact points given an image. VRB is trained from videos of humans conducting manipulation tasks. We post-process the resulting output of VRB to obtain dense pixel-wise grasp affordance codes, which are embedded as attributes in the ASK-Splat Gaussians during training. ASK-Splat adapts the concepts of a distilled feature field (DFF) [7] from the NeRF literature to the Gaussian Splatting (GSplat) model representation. ASK-Splat trains online from RGB images collected in a brief training phase prior to robot operation.

Leveraging ASK-Splat, SEE-Splat enables real-time scene-editing by extracting relevant objects from the ASK-Splat scene through a semantic 3D masking process, and subsequently moving these objects in 3D, by applying a transformation specified by a function describing the desired motion. We use SEE-Splat to reflect the movements of objects made by a robot as it carries out multi-stage manipulation tasks, so that later stages of the task can build upon the results of earlier stages (e.g., moving a saucepan to the stove, then putting a fruit in the saucepan). Moreover, SEE-Splat enables visualization of the evolution of the scene prior to the execution of the robot's plans.

Lastly, Grasp-Splat leverages the affordance and semantic codes embedded in the ASK-Splat scene to propose candidate grasp poses that are more likely to succeed. Grasp-Splat takes an open-vocabulary task command and produces 3D segmentation masks for the relevant objects using ASK-Splat's embedded semantic codes. It then queries the pre-trained grasp proposal model GraspNet [8] to produce a set of grasp candidates for the objects, and re-ranks these candidates based on alignment with the grasp affordance codes embedded in the ASK-Splat scene. Following the grasp generation, Grasp-Splat selects and executes the best grasp to accomplish the sub-task.

These components together make up Splat-MOVER, illustrated in Figure 1. Given a natural language description of a multi-stage manipulation task, Splat-MOVER utilizes the affordance-and-semantic-aware ASK-Splat scene representation to localize and mask the objects specified by the natural language prompt in the 3D scene, providing specific initial and target configurations for the robotic manipulator at each stage of the manipulation task. Subsequently, SEE-Splat edits the ASK-Splat scene in real-time, reflecting the current state of the real-world scene, thereby enabling its use in multi-stage manipulation tasks. Leveraging object-specific grasp affordance codes from ASK-Splat, Grasp-Splat generates candidate grasp configurations for each object, at each stage of the manipulation task.

We showcase Splat-MOVER's effectiveness through hardware experiments on a Kinova robot, where Splat-MOVER achieves significantly improved success rates across four single-stage open-vocabulary manipulation tasks compared to two recent baseline methods: LERF-TOGO [9] and F3RM [5]. In three single-stage manipulation tasks, Splat-MOVER improves the success rate of LERF-TOGO by a factor of at least 2.4, while achieving almost the same success rate in the last task (95% compared to LERF-TOGO's 100%). Likewise, Splat-MOVER improves the success rates of F3RM by a factor ranging from about 1.2 to 3.3, across the four single-stage manipulation tasks. In addition, we demonstrate the superior performance of Splat-MOVER in four multi-stage manipulation tasks, where we leverage SEE-Splat to reflect the updates in the scene resulting from prior manipulation stages, a capability absent in existing baseline approaches.

We summarize our contributions below:

(i) We introduce ASK-Splat, a Gaussian Splatting scene model with embeded affordance and semantic codes, enabling geometric, semantic, and affordance scene understanding.

(ii) We propose SEE-Splat, a scene-editing module that uses 3D semantic masks from ASK-Splat to enable real-time editing of Gaussian Splatting scenes, to reflect the motion of objects in the scene due to the robot's actions.

(iii) We introduce Grasp-Splat, a grasp-generation module that proposes affordance-aligned grasp candidates for specified objects, given a natural-language description of the manipulation task.

II. Related Work

In this section, we review relevant work in distilling language semantics into radiance fields and prior work in open-vocabulary scene-editing and robotic manipulation.

A. Language-Embedded NeRFs and Gaussian Splatting

Neural Radiance Fields (NeRFs) represent 3D scenes using multi-layer perceptrons (MLPs), parameterizing the color and density of each point in the scene [10]. NeRFs provide impressive high-fidelity scene renderings, even from novel views. However, to generate high-quality images, NeRFs generally require large MLPs, leading to significant training and rendering times and GPU memory usage [11], [12]. Existing NeRF methods employ dictionary/hash encodings [13], [14], and spatial discretization schemes [15], [16] to reduce the memory usage with fewer model parameters and improve the training times while maintaining or improving the rendering quality of NeRFs. Despite these advances, NeRFs still struggle to achieve real-time rendering speeds. In addition, NeRFs tend to generate *floaters* (non-existent geometry in free space), and fail to sufficiently represent free space in many scenes. Gaussian Splatting addresses these limitations of NeRFs, enabling real-time rendering [17]. Gaussian Splatting represents the environment using anisotropic 3D Gaussians, which allow for fast tile-based rasterization.

Recent advances in foundation models have facilitated grounding language or semantic features into these 3D radiance fields by distilling features extracted from pretrained foundation models, such as CLIP, to generate semantic relevancy maps through open-vocabulary queries. LERF [18] and MaskCLIP [4] propose language-embedded NeRFs that produce a 3D semantic relevancy map based on a query.

LERF embeds language features from CLIP and DINO [19] across multiple scales, while MaskCLIP modifies the model architecture of CLIP to extract denser embeddings, which are distilled into the NeRF. Recent concurrent works [20]–[23] have explored embedding language in Gaussian Splatting representations. Our contribution aligns with this nascent research area, as we incorporate language features from CLIP into Gaussian Splatting representations.

B. Open-Vocabulary 3D Scene Editing

Editable, high-fidelity scene representations are essential to the success of robotics tasks. As the robot interacts and operates within an environment, the scene representation should be updated to reflect the evolution of the scene due to interactions of the robot in the scene, especially in multi-stage tasks. Here, we present relevant prior work in 3D scene editing. CLIP-NeRF [24] presents an image or prompt-based NeRF editing method, composed of a disentangled conditional NeRF and a CLIP-driven manipulation module. In CLIP-NeRF, prompting can affect the entire scene, degrading the quality of the edited scene. Distilled Feature Fields (DFFs) [7] represents a 3D Neural Feature Field that provides query-based scene decomposition for NeRFs. DFF assigns a semantic feature to each coordinate within the 3D space. While NeRF's MLP architecture makes it challenging to decompose the scene, DFF decomposition capability enables NeRF editing by semantically selecting regions of the space. Similar to LERF [18], DFF makes use of pre-trained feature encoders from CLIP [3], DINO [19], or LSeg [25] that are trained on extensive image and text data.

Since NeRFs represent the 3D scene implicitly, scene-editing generally require costly transformations of each sampled point during the ray-tracing procedure in volumetric rendering of the scene. In our work, we leverage the explicit form of the Gaussian primitives to enable real-time scene-editing. To the best of our knowledge, this work represents the first instance of dynamic 3D-scene editing tailored for multi-stage robot manipulation.

C. Visual Affordances

Visual affordances, as defined by [1], are characteristics of objects that suggest how they might be interacted with by humans or robots. This concept has been extensively studied, with past research primarily focused on creating models that learn these affordances from images [26]–[28]. These models have been applied to various datasets, such as those involving tools [29], indoor [30], [31] and outdoor scenes [32]. Other studies, such as those by [33], [34], have explored learning visual affordances by identifying human-object interaction points in video data. However, prior work has not examined embedding grasp affordances within 3D scene representation, which is the focus of our work, for generating affordance-aware grasp configurations for manipulation tasks.

D. Open-Vocabulary Robotic Grasping and Manipulation in Radiance Fields

Traditionally, robotic manipulation methods have focused on closed-set manipulation of objects, i.e., on the manipulation of prespecified objects. However, advances in foundation models have enabled the development of open-vocabulary robotic manipulation methods, where robots manipulate objects given a natural-language query [35]. Existing open-vocabulary robotic manipulation methods rely upon semantic scene representations, which distill semantic information from vision-language foundation models, such as CLIP [3], into a NeRF representation of the scene for semantic segmentation of the scene [4], [18]. For language-guided robotic manipulation, LERF-TOGO [9] builds upon the semantic NeRF representation in LERF [18], utilizing it with a deep-learned grasp generation method GraspNet [8] for grasp generation. F3RM [36] introduces a language-guided robotic manipulation method based on human demonstrations, computing an embedding for all task-relevant demonstrations, and learns a feature field mapping each point in space to a task embedding, utilizing the NeRF-based representation in [4]. Subsequently, candidate grasps for the robot are generated through an optimization procedure.

Although these methods provide results demonstrating their effectiveness, these methods still face a number of challenges. LERF-TOGO [9] requires the specification of a grasp location on an object by a human operator or a large language model, which might not be readily available. Likewise, F3RM [36] requires human demonstrations, which might be expensive to collect. In this paper, we propose to learn the affordance of each object in the scene, obviating the need for human guidance. We distill affordance from the VRB foundation model [6], which produces an affordance map and candidate contact points from 2D images. Further, unlike existing open-vocabulary robotic manipulation methods, our proposed method utilizes a Gaussian Splatting scene representation, enabling real-time rendering and query performance.

III. Gaussian Splatting

We introduce notation relevant to this paper, before presenting Gaussian Splatting. We denote a 2D feature embedding as $f \in \mathbb{R}^{a \times b \times c}$, where the first-two dimensions (a, b) represent the spatial dimensions, while the last dimension c represents the dimension of the embedding. Similarly, we denote 1D feature embeddings as $f \in \mathbb{R}^c$. We provide a brief review of Gaussian Splatting, which we build upon to obtain a affordance-and-semantic-aware scene representation. We direct interested readers to [17] for a more in-depth discussion of Gaussian Splatting. In Gaussian Splatting, the scene is represented using 3D Gaussians, parameterized by a mean $\mu \in \mathbb{R}^3$ and covariance $\Sigma \in \mathbb{R}^{3\times3}$, with the probability density function (pdf) given by: $p(x) = \eta \exp^{-\frac{1}{2}(x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu)}$. In practice, the unnormalized probability density function is used, dropping the normalization constant $\eta \in \mathbb{R}$. Further, the covariance matrix Σ is represented as the product of a scaling (diagonal) matrix $S \in \mathbb{R}^{3\times3}$ and a rotation matrix $R \in \mathbb{R}^{3\times3}$ (given by a quaternion), expressing Σ as: $\Sigma = RSS^\mathsf{T}R^\mathsf{T}$. An opacity parameter $\alpha \in \mathbb{R}$ and spherical harmonics parameters are assigned to each Gaussian. The spherical harmonics parameters enable the Gaussians to capture the view-dependent properties of the color of each

point in space. Tile-based rasterization with α-blending is utilized for rendering the Gaussians. Specifically, the 3D Gaussians is projected to $2D$, with the projected covariance matrix given by: $\Sigma' = JW\Sigma W^{\mathsf{T}} J^{\mathsf{T}}$, where $J \in \mathbb{R}^3$ denotes the Jacobian of the projective transformation (approximated by an affine approximation) and $W \in \mathbb{R}^3$ denotes the viewing transformation. The color of each pixel C in the rendered image results from the point-based α-blending procedure:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

where N represents the number of Gaussians overlapping the pixel, c_i is computed from the spherical harmonics representation of the color of Gaussian i, and α_i in (1) denotes the opacity of Gaussian i multiplied by its unnormalized pdf (overloading the notation for the opacity parameter). The parameters of each Gaussian are optimized through stochastic gradient descent, leveraging the differentiability of the tile-based rasterization procedure to compute the gradients of the loss function $\mathcal{L}$ with respect to the parameters. In Gaussian Splatting, the loss function $\mathcal{L}$ comprises of a convex combination of the ℓ_1-photometric loss and the structural similarity index measure (SSIM) loss [37], with: $\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{SSIM}}$.

IV. Affordance-and-Semantic-Knowledge Gaussian Splatting

In this section, we present *ASK-Splat*, an Affordance-and-Semantic Knowledge Gaussian Splatting representation, that provides object-level semantic information and grasp affordance, enabling open-set queries in natural language. ASK-Splat consists of: (1) a semantic module, grounding language embeddings in a $3D$ Gaussian Splatting scene, enabling the generation of similarity (or relevancy) maps given open-vocabulary text prompts; and (2) an affordance module, grounding grasp affordance embeddings to provide highly-localized object-specific regions amenable to grasping. We illustrate these components in Figure 3, where we assume that we have a dataset $\mathcal{D}$ of RGB images, given by $\mathcal{D} = \{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$. We discuss these components in greater detail in the following subsections.

A. Grounding Language Semantics in 3D Gaussian Splatting

In NeRF-based methods, semantic knowledge is distilled into 3D via the introduction of a semantic feature field (defined by MLPs), mapping each point in both the free and occupied regions of the scene to a semantic feature embedding. However, we note that learning the semantic information over both free and unoccupied space is generally inefficient, as many human and robotics tasks only require semantic knowledge of occupied space, e.g., when asked to pick up a cup, a human typically focuses on the semantic query 'cup,' which lies within the occupied region of the scene. As a result, in ASK-Splat, we ground language embeddings in the occupied regions of the scene, represented by the 3D Gaussians.

Remark 1 (Computation Efficiency)**.** *A naive approach to encode semantic information within a Gaussian Splatting representation would be to assign an additional parameter $f \in \mathbb{R}^C$ assigned to each Gaussian, representing the semantic feature embedding associated with the Gaussian. However, such an approach introduces significant memory and computation challenges, effectively eliminating the real-time rendering rates of Gaussian Splatting. Moreover, the memory available on commercially available hardware might be insufficient for representing the scene, especially large scenes with many Gaussians.*

To preserve the real-time rendering speed of Gaussian Splatting, we learn a lower-dimensional latent space associated with the high-dimensional semantic feature embeddings, and subsequently, leverage the lower-dimensional latent features for semantic knowledge distillation, which we describe in the following discussion. Given an RGB image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, we begin by extracting 'ground-truth' image embeddings from the vision-language foundation model CLIP, denoted by $f_{\text{gt}} \in \mathbb{R}^{H_f \times W_f \times C}$, using MaskCLIP [4]. We note that although the image embeddings from [4] are sometimes described to be dense, the resulting image embeddings for the original input image are essentially patch-based, with $H_f < H$ and $W_f < W$. Further, we note that the dimension of f_{gt} (denoted by C) depends on the CLIP model used, with common values ranging between 512 (for the CLIP-ViT models) and 1024 (for the CLIP-ResNet models).

Subsequently, we learn a lower-dimensional encoding of the semantic embeddings using an autoencoder, consisting of an encoder $g_\phi^{\text{enc}} : \mathbb{R}^{H_f \times W_f \times C} \mapsto \mathbb{R}^{H_f \times W_f \times l}$, mapping semantic embeddings of dimension C from the semantic feature space to the semantic latent space, and a decoder $g_\theta^{\text{dec}} : \mathbb{R}^{H_f \times W_f \times l} \mapsto \mathbb{R}^{H_f \times W_f \times C}$, mapping the semantic latent embeddings of dimension l to the semantic feature space, with parameters ϕ and θ, respectively. We utilize feedforward MLPs in defining the encoder and decoder (depicted in Figure 3) with $l = 3 \ll C$. 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 loss function $\mathcal{L}_g$, given by:

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

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 we note that ψ applies to inputs of arbitrary height and width), and $\mathcal{D}$ denotes the dataset of images used in training the Gaussian Splatting scene, with $f_{\text{gt},i}$ denoting the ground-truth semantic features of image i. The first term in (2) represents the mean-squared-error (MSE) reconstruction loss with $\kappa_g \in \mathbb{R}_{++}$ denoting the constant

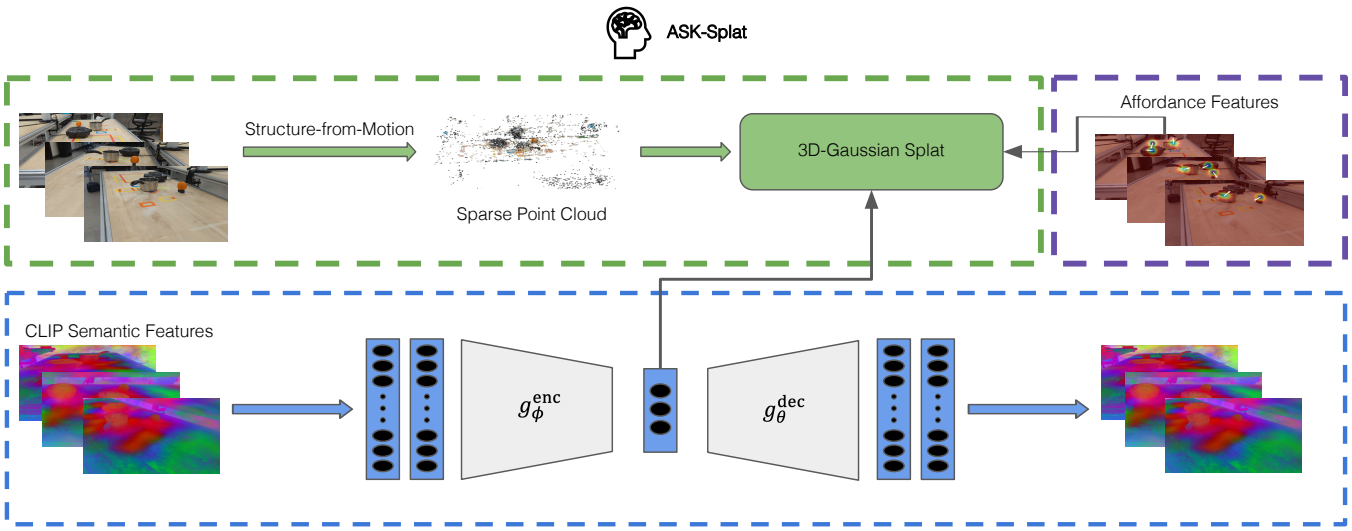

Fig. 3: ASK-Splat is trained with: (1) RGB images of the scene and is initialized with a sparse point cloud from structure-from-motion (in the green box); (2) CLIP semantic latent features computed from an encoder-decoder model (in the blue box); and (3) affordance codes from a vision-affordance foundation model (in the purple box), e.g., VRB.

associated with this term, while the second term represents the cosine-similarity loss between the ground-truth embeddings and the reconstruction.

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 representation. We augment the attributes of each Gaussian with an additional parameter $f_s \in \mathbb{R}^l$, denoting the semantic feature of the Gaussian. To render $2D$ semantic feature maps of the scene, we utilize the same tile-based rasterization procedure presented in [17], culling 3D Gaussians whose 99% confidence ellipsoid do not intersect the view frustum associated with the pose of the camera. We optimize the semantic feature parameters using the loss function:

$$\begin{aligned}\mathcal{L}_s =& \kappa_s \sum_{i=1}^{|\mathcal{D}|} \left\| \hat{\mathcal{I}}_i^f - g_\phi^{\text{enc}}(f_{\text{gt},i}) \right\|_2^2 \\ &+ \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left(1 - \psi(\hat{\mathcal{I}}_i^s - g_\phi^{\text{enc}}(f_{\text{gt},i}))\right),\end{aligned} \tag{3}$$

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 $\kappa_s \in \mathbb{R}_{++}$ denotes the constant term in the MSE loss, given by thee first component of $\mathcal{L}_s$. Although not explicitly stated in (3), we resize the output of g_ϕ^{enc} using bilinear interpolation to obtain a ground-truth semantic map of compatible dimensions.

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 Gaussian can be trained simultaneously with other spatial and visual-related parameters associated with the Gaussians, along with the autoencoder's parameters. Nonetheless, empirical evaluations suggest that a sequential training procedure in which the semantic parameters are trained after the 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 [3], [18]. 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:

$$\text{sim}(\mathcal{Q}_s, \mathcal{Q}_s') = \min_{i \in |\mathcal{Q}_s'|} \gamma(p, \nu_a(\mathcal{Q}_s), \nu_b(q_{s,i}')), \tag{4}$$

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 ith 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

(4) to distinguish feature points associated with the query from dissimilar ones.

B. Grounding Affordance in 3D Gaussian Splatting

ASK-Splat embeds object-specific grasp affordances directly within the 3D Gaussian Splatting environment, enabling real-time queries of relevant regions of an object for grasping. By distilling visual grasp affordances from vision-affordance foundation models trained on large human-object interaction datasets, such as VRB [6], into the scene representation, we endow robots with the ability to identify the parts of an object that a human is more likely to grasp, capturing common-sense human knowledge and experience in interacting with objects.

In the subsequent discussion, we present our procedure for embedding grasp affordances in ASK-Splat. We leverage the 3D Gaussian primitives in embedding $2D$ visual grasp affordances in $3D$, without resorting to a neural-based approach. We obtain the *ground-truth* affordance score by evaluating the training dataset $\mathcal{D}$ using the vision-affordance foundation model. By augmenting the attributes of each Gaussian in the scene with an additional parameter $\beta \in [0, 1]$, representing the *score* of the Gaussian affording the task of grasping, we ground the grasp affordances within the geometric primitives representing the occupied regions of the scene. We enforce the box constraints on β using the sigmoid activation function to provide smooth gradients during the training procedure.

During training, we utilize the same tile-based rasterization procedure discussed in Section IV-A in rendering the $2D$ visual grasp affordance of the scene and optimize the grasp affordance parameter β using the loss function:

$$\mathcal{L}_\beta = \kappa_\beta \sum_{i=1}^{|\mathcal{D}|} \left\| \hat{\mathcal{I}}_i^\beta - \mathcal{I}_i^\beta \right\|_2^2, \tag{5}$$

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.

V. Scene-Editing-Enabled Gaussian Splatting

We present SEE-Splat, our module for Scene-Editing-Enabled Gaussian Splatting representations. SEE-Splat harnesses the semantic and affordance knowledge in ASK-Splat to identify and localize relevant objects within a scene for insertion, removal, or modification of the object's visual or spatial properties. SEE-Splat consists of three components: (i) a semantic component utilizing natural-language queries to generate a relevancy map based on semantic similarity; (ii) a Gaussian masking module that generates a sparse point cloud of the relevant objects, comprising of all semantically-relevant Gaussians; and (iii) a scene transformation component that edits the 3D scene by inserting, removing, or modifying the geometric, spatial, or visual properties of the Gaussians. In Figure 4, we illustrate each component in editing a real-world Kitchen scene to visualize the action of moving the saucepan from the table to the electric stove, showing localization of a saucepan, extraction of the semantically-relevant Gaussians, and transformation of the relevant Gaussians. Together, these components enable robots to visualize the effects of their interactions with other objects in a virtual 3D scene, prior to executing these actions in the real-world. Essentially, SEE-Splat provides a digital twin of the scene. Moreover, provided sensor feedback is available, SEE-Splat enables real-time visualization of the real-world in a virtual environment, enabling the Gaussian primitives to accurately reflect the real-time geometry and visual properties of objects in the real-world. We discuss each component in the following subsections.

A. 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 IV-A. Figure 4 shows the localization of an electric stove, saucepan, and a fruit in a real-world Kitchen 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.

B. Masking the Gaussians in SEE-Splat

Given a semantic similarity map of the scene, SEE-Splat generates a mask identifying the Gaussians relevant to the specified object. This procedure begins with thresholding the semantic scores of each Gaussian to remove dissimilar Gaussians from the set of relevant Gaussians, which creates a sparse point cloud of the relevant Gaussians, constructed from the means of these Gaussians. However, photo-realistic rendering of Gaussian environments require denser point clouds. Consequently, SEE-Splat lifts the features of the point cloud from the 3D Euclidean space to a 7D feature space, by augmenting each point in the point cloud with its RGB color and semantic score. Subsequently, SEE-Splat identifies all neighboring points in the scene within a specified distance of the point cloud in the 7D feature space using an efficient KD-tree query. SEE-Splat incorporates these points into the point cloud to create a denser point cloud, comprising of all semantically-relevant Gaussians, while removing outliers from the set of points. In Figure 4, we show the Gaussians extracted by SEE-Splat, as a point cloud with well-defined geometry, given a natural-language query for each object.

C. Editing the Gaussians in SEE-Splat

Leveraging the Gaussian primitives in ASK-Splat, SEE-Splat enables real-time scene-editing by inserting new Gaussians into the scene, removing Gaussians, and modifying the properties of the Gaussians, to reflect (or simulate) changes in the real-world. SEE-Splat supports seamless

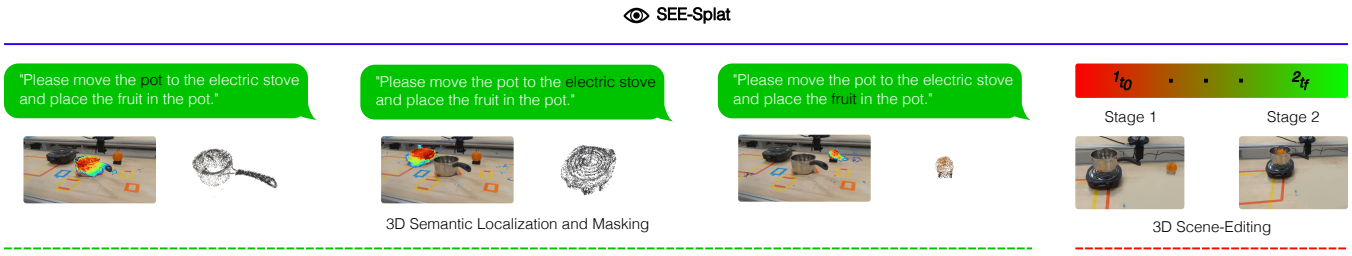

Fig. 4: Given a natural-language prompt and a desired scene transformation, SEE-Splat leverages 3D ASK-Splat for open-vocabulary scene-editing via semantic localization of relevant Gaussian primitives in the scene, followed by 3D masking and transformation of these Gaussians. We visualize the Gaussians as as a point cloud, with well-defined geometry.

insertion and removal of Gaussians by introducing or deleting the relevant Gaussians from the set of Gaussians representing the scene, respectively. In addition, SEE-Splat supports both rigid and non-rigid transformation of the Gaussians, enabling simulated motion of the Gaussians, as well as changes to the shape of the Gaussians via non-isometric scaling. Specifically, given a function specifying the transformation $\xi : \mathcal{G}_s \mapsto \mathcal{G}_s$ (where $\mathcal{G}_s$ represents the space of the Gaussian primitives), SEE-Splat updates the spatial attributes of the relevant Gaussians by applying ξ to these Gaussians. In the case of rigid transformations, ξ can be described by an $\mathrm{SE}(3)$ transformation matrix, specifying rotation and translation of the Gaussians. We can render the edited scene to generate photo-realistic visualizations. Although, we do not consider physics-based simulations in this work, we note that physics can be incorporated into SEE-Splat to achieve realism. We expound on this point in Section IX.

Deletion and transformation of the Gaussians introduces artifacts into the scene representation, degrading its photo-realistic qualities. To address this challenge, SEE-Splat enables 3D Gaussian infilling by introducing new Gaussians with similar attributes in regions with missing geometry. Figure 5 provides an illustration of such artifacts (e.g., the hole in the table), when the scene is edited to visualize the effects of moving the saucepan to the electric stove. Through 3D Gaussian infilling, SEE-Splat generates a photorealistic rendering of the edited scene, eliminating these artifacts.

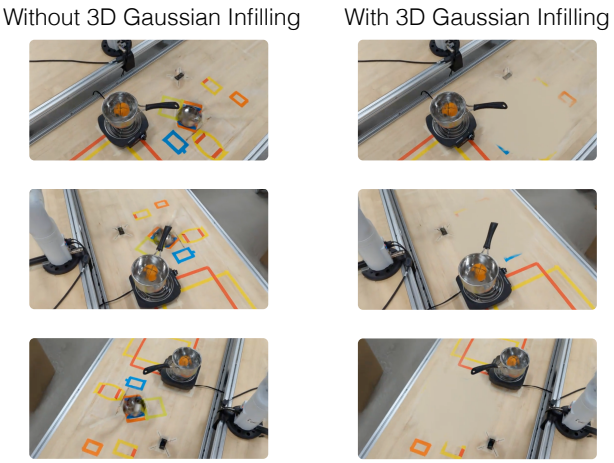

Fig. 5: 3D Gaussian Infilling in SEE-Splat: (Left) In general, without 3D Gaussian infilling, transformation of the Gaussians (e.g., moving the saucepan from the table to the electric stove) introduces artifacts, such as the hole in the table after moving the saucepan. (Right) Via 3D Gaussian infilling, SEE-Splat generates photorealistic renderings of the edited scene.

VI. Grasping and Manipulation with Splat-MOVER

Via Splat-MOVER, we provide a concrete application of ASK-Splat and SEE-Splat to multi-stage robotic manipulation problems. We consider *language-guided*, *multi-stage* robotic grasping and placing tasks, where each stage consists of grasping and placing distinct objects within a scene, which arise, for example, in scenarios where robotic manipulators are utilized in meal preparation or cleanup. Specifically, given a language description of a task, we seek to design an algorithm that endows a robotic manipulator with spatial and semantic understanding of its environment to enable it to complete a specified grasping and placing task, while incorporating the temporal effects of its actions in its representation of its environment to facilitate the completion of downstream stages. We assume that the manipulator is equipped with a parallel-jaw gripper; however, we note that the algorithm in this work can be extended to other gripper forms. Building upon ASK-Splat and SEE-Splat, Splat-MOVER introduces Grasp-Splat, a grasp generation and selection module for robotic manipulation, described in the following discussion.

A. Grasp-Splat for Grasp Proposal

Grasp-Splat utilizes the dense point cloud of the object generated by SEE-Splat (e.g., Figure 4) to propose grasp configurations for grasping the object, while harnessing the grasp affordance knowledge in ASK-Splat to identify more compatible grasp configurations associated with the specified object. The proposed grasps are generated from the point cloud using GraspNet [8], a learning-based approach that utilizes an encoder-decoder architecture to estimate $6D$ grasp poses for a parallel-jaw gripper from a point cloud of the object, along with estimated grasp-quality scores associated with these grasp poses.

We note that the grasps generated by GraspNet are not always ideal. For example, the grasps generated by GraspNet in Figure 6 are either infeasible or challenging to execute. Consequently, Grasp-Splat executes a grasp selection

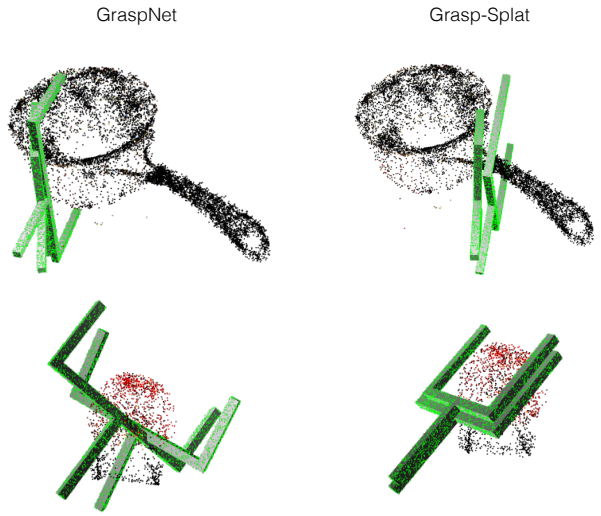

Fig. 6: The top-two candidate grasps proposed by (left) GraspNet and (right) Grasp-Splat, leveraging the grasp affordance of each object. Qualitatively, the grasps generated by Grasp-Splat are more likely to succeed, compared to the grasps generated solely from Grasp-Net. Further, the grasps proposed by Grasp-Splat are better localized on the handle of the saucepan.

procedure to identify more-promising candidate grasps. We hypothesize that leveraging grasp affordance of each part of the object when generating candidate grasps could be essential to identifying better candidate grasps. As a result, we introduce the grasp metric $\nu : \mathrm{SE}(3) \mapsto \mathbb{R}$, which computes the grasp affordance at a specified grasp pose $X \in \mathrm{SE}(3)$. Grasp-Splat ranks the candidate grasps generated by GraspNet based on the grasp scores given by $\nu(X)$, leveraging the affordance associated with each grasp pose to identify grasp configurations that are more likely to succeed, depicted in Figure 6. Moreover, since GraspNet does not consider the position of the robot relative to the object, the proposed grasps might require post-processing to account for the relative position of the robot.

B. Multi-Stage Robotic Manipulation

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

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

VII. Evaluations

We examine the effectiveness of ASK-Splat, SEE-Splat, and Splat-MOVER in open-vocabulary, multi-stage robotic manipulation problems. We begin with evaluating the affordance and semantic knowledge embedded in ASK-Splat, assessing the quality of the encoded grasp affordance and semantic masks generated by ASK-Splat across a variety of environments and objects.

A. Experimental Setup

1) ASK-Splat: We distill grasp affordances from the vision-affordance foundation model VRB [6], which is trained on the EPIC-KITCHENS dataset [38], consisting of videos of humans performing kitchen tasks, such as cutting fruits and vegetables. We note that the generalization of the affordance knowledge in ASK-Splat is limited by that of VRB, the underlying foundation model. VRB utilizes Language Segment-Anything (LangSAM) [39], which requires the specification of objects within each image for which it predicts the contact locations and motion direction after contact. This requirement is not limiting, in practice, as end-to-end object detectors that provide bounding boxes for all objects in the scene [40]–[42] could be used. We distill the grasp affordance scores from the heatmaps computed by VRB and the semantic embeddings from the vision-language foundation model `RN50x64`, CLIP-ResNet model [9]. We implement ASK-Splat in Nerfstudio [43]. To train ASK-Splat, we record a video of each scene using a smartphone and utilize the training API available in Nerfstudio, using the sparse point cloud computed via structure-from-motion [44] for initialization.

2) Scenes: We consider only real-world scenes in our experiments, including a *Kitchen* scene (consisting of common kitchen cookware such as saucepans, chopping boards, and knives); *Cleaning* scene (consisting of common household

cleaners such as disinfectant wipes, dish soaps, and cleaning sprays); *Meal* scene (consisting of cutlery such as plates, spoons, forks, and cups); *Random* scene (consisting of random items such as a pair of scissors, chess pieces, and keyholders); and a *Workshop* scene (consisting of tools such as a power drill, work mat, and scraper). Figure 8 shows these scenes. We note that the *Workshop* and *Random* scenes contain out-of-distribution objects with respect to the EPIC-KITCHENS dataset (i.e., objects not found in a typical kitchen), such as the power drill and the chess pieces.

3) Splat-MOVER: We consider the multi-stage robotic manipulation task where the robot must sequentially pick and place two different objects and move them to a common goal location. The task is specified by a user that provides an open-vocabulary command, e.g., "Pick up the saucepan and move it to the burner, then pick up the lid and put it on the saucepan." For simplicity, we limit the task to two sequential pick-and-place maneuvers. However, we note that Splat-MOVER does not impose this limitation 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. We show novel views of the scene, rendered from ASK-Splat, associated with each task in Figure 7.

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 [45] 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.

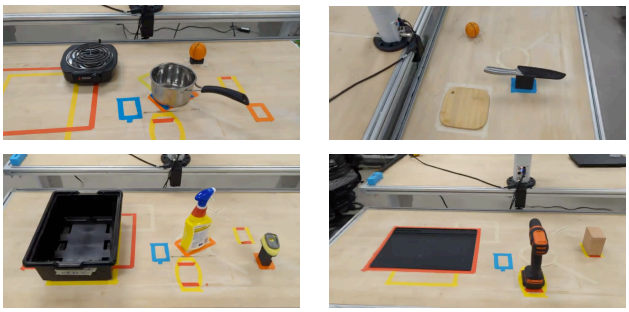

Fig. 7: Multi-stage Robotic Manipulation: (top-left) *Cooking* task, where a robot is asked to move a saucepan to an electric burner, followed by placing a fruit in the saucepan; (top-right) *Chopping* task, where a robot is asked to move a knife to a chopping board, followed by placing a fruit next to the knife; (bottom-left) *Cleaning* task, where a robot is asked to move a cleaning spray into a bin, followed by placing a sponge next to the bin; (bottom-right) *Workshop* task, where a robot is asked to move a power drill to a work mat, followed by moving a wooden block next to the drill.

B. 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 8, 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 8, 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 $0.592 \pm 7.20\text{e}^{-2}$, 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 8 (bottom row). Likewise, the Meal scene achieves the highest SSIM score of $0.681 \pm 8.91\text{e}^{-2}$, 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 $0.648 \pm 9.06\text{e}^{-2}$, $0.647 \pm 1.30\text{e}^{-1}$, and $0.614 \pm 8.37\text{e}^{-2}$, respectively.

Figure 9 shows the semantic masks generated by ASK-Splat across different scenes. Given a natural-language

Fig. 8: 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).

query, ASK-Splat localizes the relevant object in the scene based on the cosine-similarity of the Gaussians to the query. In Figure 9, 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 9. 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.

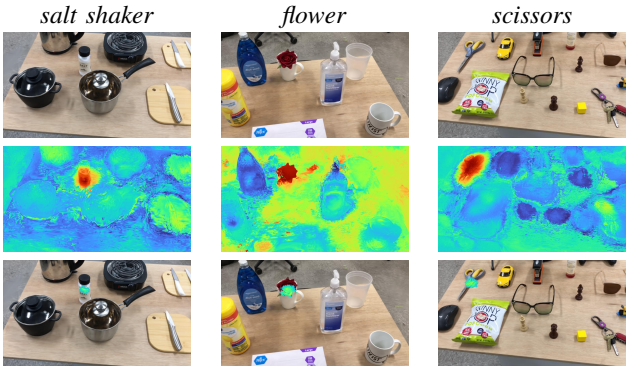

Fig. 9: Affordance and Language Semantics in ASK-Splat: Given natural-language queries, ASK-Splat renders: (top-row) RGB images, (middle-row) semantic masks of the scene, and (bottom-row) localized grasp affordance regions, for example, for a *salt shaker* in the *kitchen* scene, a *flower* in the *cleaning* 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.

C. Splat-MOVER for Multi-Stage Robotic Manipulation

We compare Splat-MOVER to the existing open-vocabulary robotic manipulation methods LERF-TOGO [9] and F3RM [5] in four tasks: the *Cooking* task, *Chopping* task, *Cleaning* task, and *Workshop* task, described in Section VII-A.3. We utilize the publicly-available implementation of LERF-TOGO provided by the authors. Since F3RM requires collecting human demonstrations in virtual-reality, which we could not provide in our evaluations, we implement F3RM and utilize GraspNet for grasp generation. Since the NeRF-based methods LERF-TOGO and F3RM are not amenable to scene-editing, we limit our comparisons to the first stage of the manipulation task, where we consider grasping a saucepan, knife in a knife guard, cleaning spray, and power drill. Figure 10 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 I.

TABLE I: Representation Capabilities of LERF-TOGO [9], F3RM [5], and Splat-MOVER

Capabilities	Semantic Knowledge	Affordance Knowledge	Scene-Editing
LERF-TOGO [9]	✓	✗	✗
F3RM [5]	✓	✗	✗
Splat-MOVER (**ours**)	✓	✓	✓

In the hardware experiments on the robot, we describe a candidate grasp as being feasible if the MoveIt planner successfully finds a plan to execute the grasp. We execute 20 feasible trials for each object, and in Table II, provide the grasping success rate of each method, as well as the percentage of grasps that lie within the affordance region of the object (AGSR), where the affordance region of each object is defined as follows: the entire handle of the saucepan, the entire knife with the guard on, the region of the cleaning spray excluding its cap, and the middle region of the power drill below its top compartment and above its battery compartment. From Table II, Splat-MOVER achieves the best grasping success rates and the highest percentage of grasps in the affordance region of each object in grasping the saucepan, knife, and cleaning spray. LERF-TOGO attains a perfect success rate in grasping the power drill; however, LERF-TOGO always grasped the top of the drill, outside its affordance region, depicted in Figure 10. In contrast, Splat-MOVER achieves a 95% success rate, grasping the object within its affordance region.

In addition, we evaluate the pick-and-place success rate of all the methods in each manipulation task, where the place success rate is conditioned on the number of successful trials in picking the object. The publicly-available implementation of LERF-TOGO does not support the specification of a place location. Hence, we do not evaluate its place success rate. In Table III, we present the pick-and-place success rates in the Cooking task. Splat-MOVER achieves a perfect success rate in picking up the saucepan, with a place success rate of 60%. We note that the place step involves placing the saucepan on an electric burner, which is quite challenging, potentially explaining the lower place success rate. Nonetheless, Splat-MOVER outperforms LERF-TOGO and F3RM in the first stage of the task, the only stage of the task that both methods can perform.

Table IV provides the pick-and-place success rates in the Chopping task. Again, 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 success rates in the second stage of each task, shown in Tables V and VI. 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.

VIII. Conclusion

We present Splat-MOVER, a robotics stack for multi-stage open-vocabulary robotic manipulation. Splat-MOVER consists of three modules: ASK-Splat, SEE-Splat, and Grasp-Splat. ASK-Splat enables semantic and affordance queries via natural-language interaction to identify relevant objects within 3D scenes, while SEE-Splat provides real-time, dynamic scene editing, enabling visualization of the evolution of the real-world scene due to the robot's interaction with objects within the scene. Grasp-Splat builds upon these two modules for affordance-aware grasp generation, necessary for effective multi-stage robotic manipulation. We demonstrate the effectiveness of Splat-MOVER in real-world experiments in comparison to two recent baseline methods. Compared to the existing works, Splat-MOVER endows robots with the unique capability for multi-stage, open-vocabulary manipulation with minimal human inputs, by leveraging distilled grasp affordance knowledge and real-time dynamic scene editing, which are essential to multi-stage manipulation.

IX. Limitations and Future Work

We present a few limitations of Splat-MOVER and provide directions for future work. ASK-Splat distills grasp affordance knowledge from foundation models into 3D Gaussian Splatting scenes. We note that the effectiveness of the distilled grasp affordances and the generalization capability of our model is highly dependent on that of the affordance foundation model. For example, when trained using the VRB foundation model, an ASK-Splat scene may not generalize notably well to markedly-different objects outside of those seen in the EPIC-KITCHENS dataset. Future work will examine enhancing the generalization capability of the proposed ASK-Splat module by training and employing diverse cross-environment, cross-task, vision-affordance foundation models, rather than relying on VRB, which is specifically trained for kitchen tasks. Further, in its current form, the distilled visual grasp affordance does not depend on the orientation of the candidate grasp, limiting Grasp-Splat from fully harnessing grasp affordances in identifying better grasp configurations in $\mathrm{SE}(3)$. Future work will consider methods for extending the computation of grasp affordance to $\mathrm{SE}(3)$. Moreover, future

Fig. 10: 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 II: Grasping success rate and the percentage of grasps in the affordance region (AGSR) of each object for LERF-TOGO [9], F3RM [5], and Splat-MOVER across 20 feasible trials. The top-performing stat is shown in bold.

Methods	Saucepan		Knife		Cleaning Spray		Powerdrill	
	Grasping Success (%)	AGSR (%)	Grasping Success (%)	AGSR (%)	Grasping Success (%)	AGSR (%)	Grasping Success (%)	AGSR (%)
LERF-TOGO [9]	40	5	35	35	25	15	**100**	0
F3RM [5]	30	30	60	60	75	40	70	70
Splat-MOVER (**ours**)	**100**	**60**	**85**	**85**	**90**	**90**	95	**95**

work will seek to integrate fast online Gaussian Splatting into ASK-Splat, eliminating the need for a brief scanning phase of the scene prior to training in addition to speeding up the training procedure. Additionally, we will seek to

TABLE III: Pick and Place Success Rates in a two-stage manipulation Cooking task, where the robot must move a saucepan to an electric burner (Stage 1), then move a fruit into the saucepan (Stage 2).

Methods	Stage 1		Stage 2	
	Pick Success (%)	Place Success (%)	Pick Success (%)	Place Success (%)
LERF-TOGO [9]	40	N/A	N/A	N/A
F3RM [5]	30	0	N/A	N/A
Splat-MOVER (**ours**)	**100**	**60**	50	80

TABLE IV: 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 [9]	35	N/A	N/A	N/A
F3RM [5]	60	**91.67**	N/A	N/A
Splat-MOVER (**ours**)	**85**	82.35	65	100

TABLE V: 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 [9]	25	N/A	N/A	N/A
F3RM [5]	75	6.67	N/A	N/A
Splat-MOVER (**ours**)	**90**	**83.33**	70	100

TABLE VI: 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 [9]	100	N/A	N/A	N/A
F3RM [5]	70	7.14	N/A	N/A
Splat-MOVER (**ours**)	**95**	**94.74**	85	88.24

integrate sensor feedback into SEE-Splat to enable closed-loop, real-time scene editing, improving the accuracy of the scene representation. In addition, by closing the scene-editing loop with sensor feedback, we can ensure that our 3D scene representation updates dynamically according to unexpected real-world events. For example, if the pan slips out of the robot gripper, SEE-Splat will be able to promptly reflect this change and modify the plans of the robot accordingly.

REFERENCES

[1] J. J. Gibson, "The ecological approach to the visual perception of pictures," *Leonardo*, vol. 11, no. 3, pp. 227–235, 1978.

[2] D. A. Norman, *The psychology of everyday things.* Basic books, 1988.

[3] 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)*. PMLR, 2021, pp. 8748–8763.

[4] C. Zhou, C. C. Loy, and B. Dai, "Extract free dense labels from CLIP," in *European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 696–712.

[5] 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.

[6] 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)*, 2023, pp. 13 778–13 790.

[7] S. Kobayashi, E. Matsumoto, and V. Sitzmann, "Decomposing nerf for editing via feature field distillation," in *Advances in Neural Information Processing Systems*, vol. 35, 2022. [Online]. Available: https://arxiv.org/pdf/2205.15585.pdf

[8] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.

[9] 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)*. PMLR, 2023, pp. 178–200.

[10] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[11] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5470–5479.

[12] ——, "Zip-nerf: Anti-aliased grid-based neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 697–19 705.

[13] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM transactions on graphics (TOG)*, vol. 41, no. 4, pp. 1–15, 2022.

[14] T. Takikawa, A. Evans, J. Tremblay, T. Müller, M. McGuire, A. Jacobson, and S. Fidler, "Variable bitrate neural fields," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–9.

[15] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510.

[16] X. Wu, J. Xu, Z. Zhu, H. Bao, Q. Huang, J. Tompkin, and W. Xu, "Scalable neural indoor scene rendering," *ACM transactions on graphics*, vol. 41, no. 4, 2022.

[17] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[18] 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)*, 2023, pp. 19 729–19 739.

[19] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9630–9640. [Online]. Available: https://doi.org/10.1109/ICCV48922.2021.00951

[20] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, "Langsplat: 3d language gaussian splatting," *arXiv preprint arXiv:2312.16084*, 2023.

[21] X. Hu, Y. Wang, L. Fan, J. Fan, J. Peng, Z. Lei, Q. Li, and Z. Zhang, "Semantic anything in 3d gaussians," *arXiv preprint arXiv:2401.17857*, 2024.

[22] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. Kadambi, "Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields," *arXiv preprint arXiv:2312.03203*, 2023.

[23] X. Zuo, P. Samangouei, Y. Zhou, Y. Di, and M. Li, "Fmgs: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding," *arXiv preprint arXiv:2401.01970*, 2024.

[24] C. Wang, M. Chai, M. He, D. Chen, and J. Liao, "CLIP-NeRF: Text-and-image driven manipulation of neural radiance fields," in *CVPR*, 2022, pp. 3835–3844.

[25] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven semantic segmentation," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=RriDjddCLN

[26] H. O. Song, M. Fritz, D. Goehring, and T. Darrell, "Learning to detect visual grasp affordance," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 798–809, 2015.

[27] P. Ardón, E. Pairet, R. P. Petrick, S. Ramamoorthy, and K. S. Lohan, "Learning grasp affordance reasoning through semantic relations," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4571–4578, 2019.

[28] N. Yamanobe, W. Wan, I. G. Ramirez-Alpizar, D. Petit, T. Tsuji, S. Akizuki, M. Hashimoto, K. Nagata, and K. Harada, "A brief review of affordance in robotic manipulation research," *Advanced Robotics*, vol. 31, no. 19-20, pp. 1086–1101, 2017.

[29] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1374–1381.

[30] A. Roy and S. Todorovic, "A multi-scale CNN for affordance segmentation in RGB images," in *14th European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 186–201.

[31] M. Hassanin, S. Khan, and M. Tahtali, "Visual affordance and function understanding: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–35, 2021.

[32] C. Pohl, K. Hitzler, R. Grimm, A. Zea, U. D. Hanebeck, and T. Asfour, "Affordance-based grasping and manipulation in real world applications," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9569–9576.

[33] T. Nagarajan, C. Feichtenhofer, and K. Grauman, "Grounded human-object interaction hotspots from video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8688–8697.

[34] M. Goyal, S. Modi, R. Goyal, and S. Gupta, "Human hands as probes for interactive object understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 3293–3303.

[35] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023.

[36] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola, "Distilled feature fields enable few-shot language-guided manipulation," *arXiv preprint arXiv:2308.07931*, 2023.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[38] 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)*, 2018, pp. 720–736.

[39] L. Medeiros, "Language Segment-Anything," https://github.com/luca-medeiros/lang-segment-anything, 2023.

[40] 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*, 2021, pp. 7373–7382.

[41] 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*. Springer, 2020, pp. 213–229.

[42] 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.

[43] 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.

[44] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] 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*, vol. 5(1), p. 3–16, 2014.