SceneFun3D: Fine-Grained Functionality and Affordance Understanding in 3D Scenes

Supplementary Material

Alexandros Delitzas[1] Ayça Takmaz[1] Federico Tombari[2,3] Robert Sumner[1]
Marc Pollefeys[1,4] Francis Engelmann[1,2]
[1]ETH Zurich [2]Google [3]TUM [4]Microsoft

This supplemental complements the main paper, by providing details of our web annotation framework, additional dataset statistics and data acquisition details. We also include implementation details for our baseline models and discuss potential applications and limitations.

1. Annotation framework

To facilitate the data collection and semantic annotation on large and high-resolution point clouds, we implement an annotation framework consisting of a number of interfaces. Our annotation pipeline consists of three stages: 1) functionality annotations, 2) natural language task descriptions collection, 3) motion annotations. Each annotation stage is followed by a corresponding verification step.

For the laser scan management interface and for storing the annotations, we use React.js[1] and MongoDB[2]. For the annotation process, we develop 3D interactive interfaces using Three.js[3] which is based on WebGL. Our interfaces can run on a simple web browser.

To enable the semantic annotation of large and high-resolution point clouds while keeping the computational requirements low, we utilize an accelerated ray-casting algorithm based on Bounding Volume Hierarchies (BVH) [5]. The 3D points are automatically grouped into recursive bounding volumes constructing a KD-tree, as can be seen in Fig. 1. As a result, spatial queries triggered by the annotator's clicks during annotation are significantly accelerated by searching on the tree nodes in a top-down manner instead of naively iterating across all the 3D points in the scene.

For each task, detailed instructions are provided to annotators. The annotation interactive interfaces for functionality annotation, natural language task description collection and motion annotation can be seen in Figures 2, 3 and 4 respectively. The annotations are verified and refined by human verifiers. For verification, all the available annotations for a scene are displayed at the interface and the human verifiers can choose to "edit" an existing annotation or "delete" it.

[1]https://react.dev
[2]https://www.mongodb.com
[3]https://threejs.org

2. The SceneFun3D dataset

2.1. Cropping the laser scans

We observe that the Faro laser scans may contain extraneous points out of the indoor scene context due to transparent surfaces, such as windows. This can significantly increase the size of the 3D point cloud with points that are not useful for our proposed tasks. To this end, we provide masks to filter these points and keep only the main scene region. Specifically, as a final step after we perform the registration described in Sec. 4.3 of the main paper, we utilize the ARKit mesh reconstructions based on the available video sequences as a reference to accurately crop the laser scans. The algorithm that we use to process each laser scan can be seen in Algorithm 1.

2.2. Constructing the scenes

To complement the main paper, we provide an overview of the main steps performed to construct the scenes in the SceneFun3D dataset in Fig. 5. Next, we utilize the processed high-resolution laser scans in our annotation pipeline.

2.3. Annotation statistics

Our work amounts to the total of 675h annotation time and 237h of verification time. Annotation took on average 57m per scene (35m for functionality annotation, 10m for the language descriptions collection and 12m for the 3D motion annotations). Verification took on average 20m per scenes (10m for functionalities, 5m for language descriptions and 5m for motions).

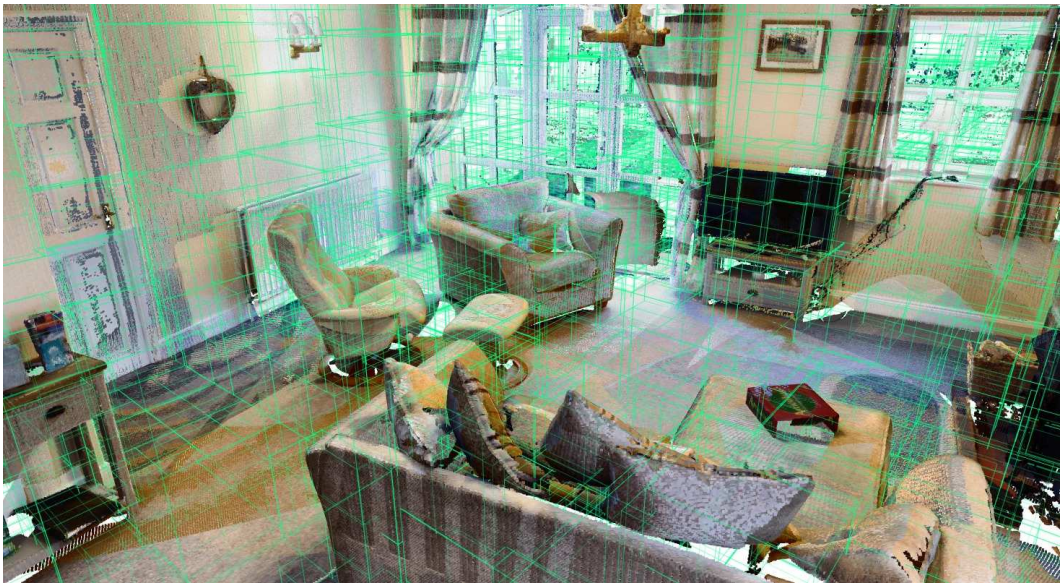

Figure 1. Our annotation interface utilizes an accelerated ray-casting algorithm where 3D points are grouped based on Bounding Volume Hierarchies (BVH). This enables the annotation of large and high-resolution point clouds with minimum computational requirements.

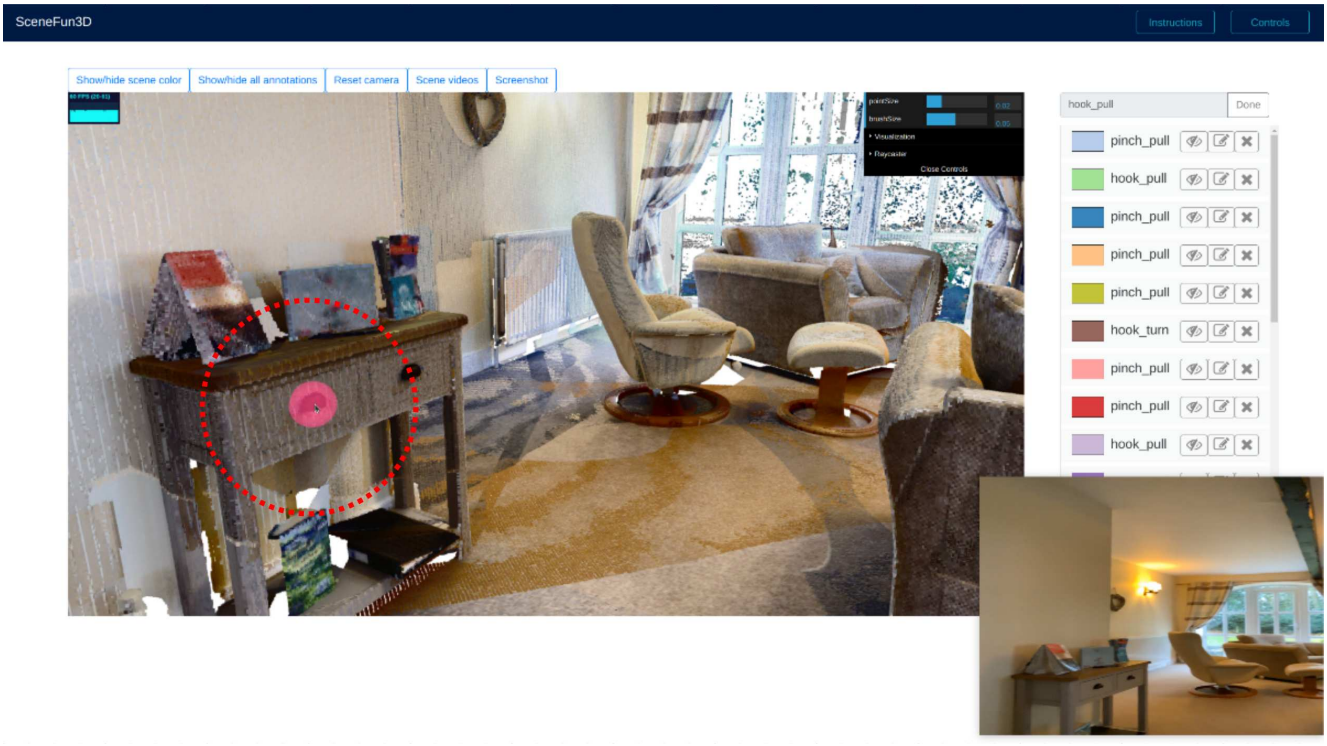

Figure 2. Our interactive web interface for functionality annotation. The user explores the 3D scene with our intuitive interactive controls. When the user detects a functional interactive element in the scene (denoted by the red-dashed circle in the figure), they first add the corresponding affordance category (e.g., hook pull) and then use the mouse to select the points that belong to the corresponding affordance mask. The brush size can be manually adjusted to facilitate the annotation of larger/smaller areas. During the annotation process, the user can also view a video sequence of the scene (bottom right part).

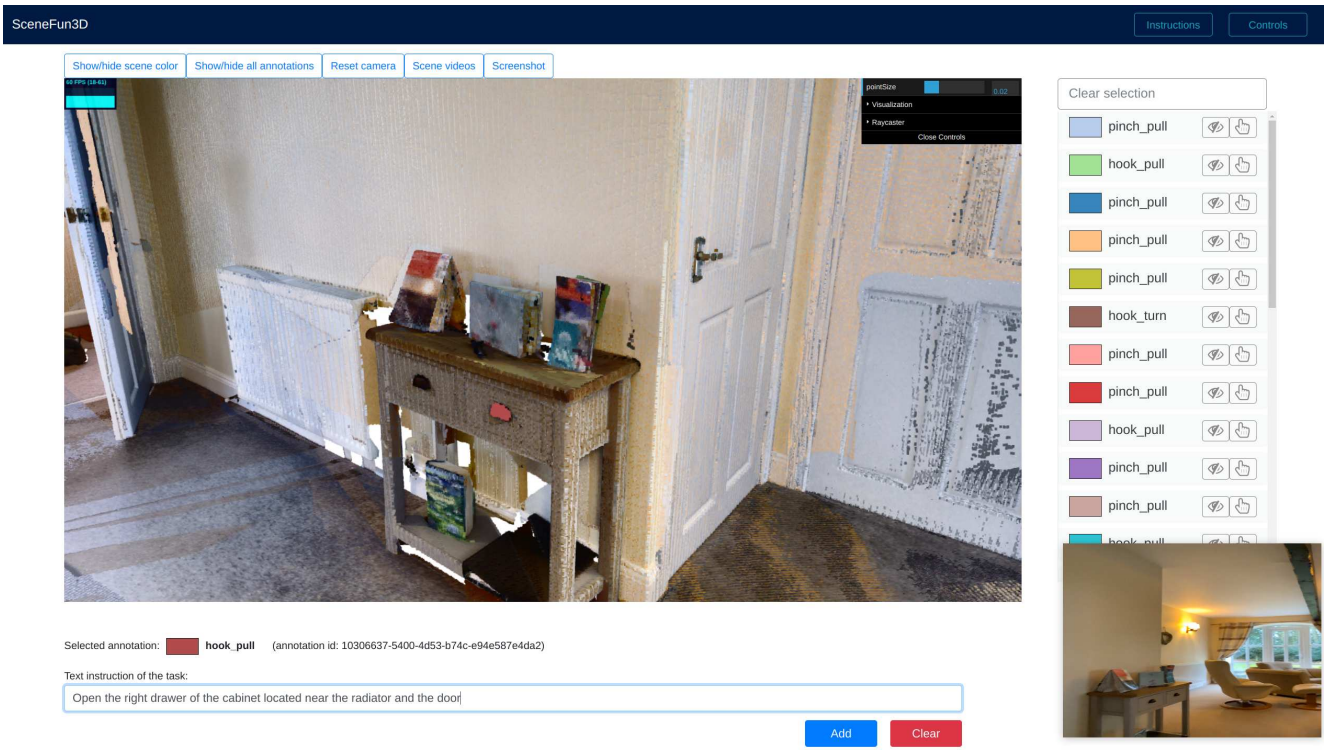

Figure 3. Our interactive web interface for the collection of natural language task descriptions. The annotator selects a functionality annotation from the menu on the right and adds a corresponding text instruction in the text box.

Algorithm 1: Cropping the laser scans

Input : laser point cloud P of N points, number of the available ARKit mesh reconstructions K for the scene, ARKit mesh reconstructions $\{M_i\}_{i=1}^{K}$, transformations to register the laser point cloud in the coordinate system of each ARKit mesh reconstruction $\{T_i\}_{i=1}^{K}$

Output: Cropped laser point cloud P^c

$C \leftarrow$ initialize as a list of N elements

for $j = 1, 2, ..., N$ **do**

  $C[j] \leftarrow False$

end

for $i = 1, 2, ..., K$ **do**

  $B_i \leftarrow$ calculate the oriented bounding box of M_i

  $B_i^P \leftarrow$ apply T_i^{-1} on B_i

  for $j = 1, 2, ..., N$ **do**

    if $P[j]$ *is in* B_i^P **then**

      $C[j] \leftarrow True$

  end

end

$P^c \leftarrow$ filter P based on the mask C

return P^c

2.4. Additional data statistics

In Fig. 6, we present the distribution of the affordance categories in our dataset. On average, each scene comes with 20.9 functionality annotations, 24.1 language task descriptions (15.4 collected from human annotators and 8.7 rephrased with ChatGPT) and 20.1 motion annotations. The average length of the language task descriptions is 7.9 words.

2.5. Natural language task description augmentation

To augment our dataset, we rephrase the collected description providing the ChatGPT model *gpt-3.5-turbo-instruct* with the following prompt: "Help me reword a sentence to a different format but keep its meaning". Below, we show examples of the original language task descriptions (**O**) collected with the help of human annotators and the rephrased (**R**) ones that we obtain:

- **O**: Turn on the TV using the remote control.
- **R**: Use the remote control to turn on the TV.

- **O**: Adjust the volume of the stereo system.
- **R**: Modify the stereo system's volume.

- **O**: Close the door.
- **R**: Shut the door.

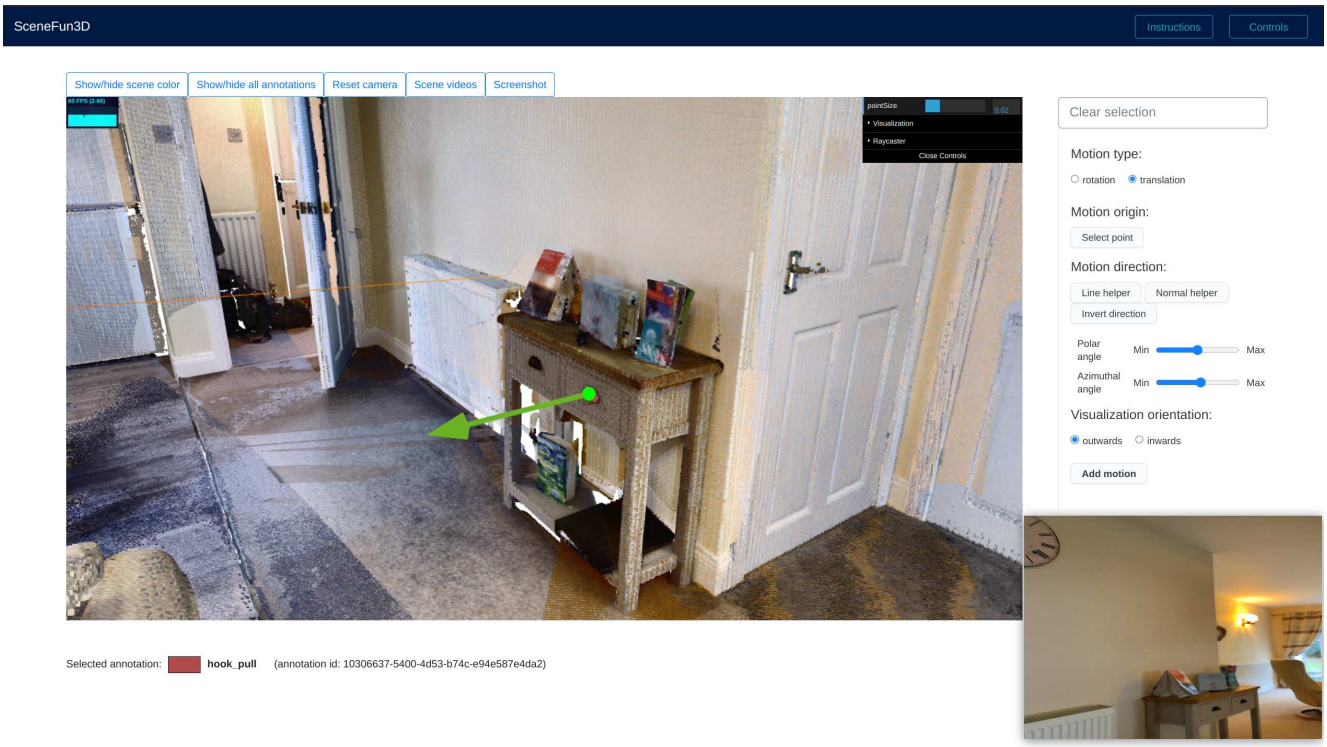

Figure 4. Our interactive web interface for the collection of motion annotations. The annotator first selects a functionality annotation to add motion parameters and then uses the helper tools to select the motion type (translational or rotational), axis of motion and motion origin (only needed for rotation).

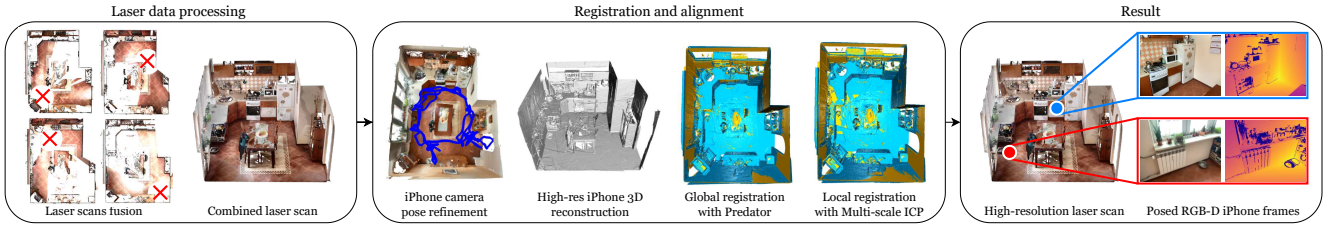

Figure 5. Overview of the main steps performed to construct the scenes in the SceneFun3D dataset.

- **O**: Open the rightmost cabinet door of the TV stand.
- **R**: Open the TV stand's rightmost cabinet door.
- **O**: Open the bottom drawer of the cabinet.
- **R**: Retrieve the contents from the bottom drawer of the cabinet.

2.6. Additional annotation examples

Additional annotation examples are illustrated in Fig. 7.

3. Implementation details

3.1. Models for functionality segmentation

Mask3D-F. For the task of functionality segmentation we modify the Mask3D [7] model, as described in the main paper. Here, we provide implementation details to ease reproducibility. We train for 1000 epochs on an NVIDIA A100 GPU using AdamW [4] and a one-cycle learning rate schedule with a maximal learning rate of 0.0001. We set the batch size to 2 and train on 2 cm voxelization to retain high-resolution details. We optimize each predicted mask using Dice loss, as it is specifically designed to address data imbalances. The overall training loss is formulated as $\mathcal{L}_{\text{seg}} = \lambda_{\text{dice}}\mathcal{L}_{\text{dice}} + \lambda_{\text{ce}}\mathcal{L}_{\text{ce}}$, where we experimentally set $\lambda_{\text{dice}} = 5$ and $\lambda_{\text{ce}} = 2$. The transformer decoder in our Mask3D-F pipeline starts with 80 instance queries. We employ curriculum masking with an initial expansion radius $r_0 = 0.1$, a decay rate $d = 0.5$ and a decay interval $\alpha = 200$.

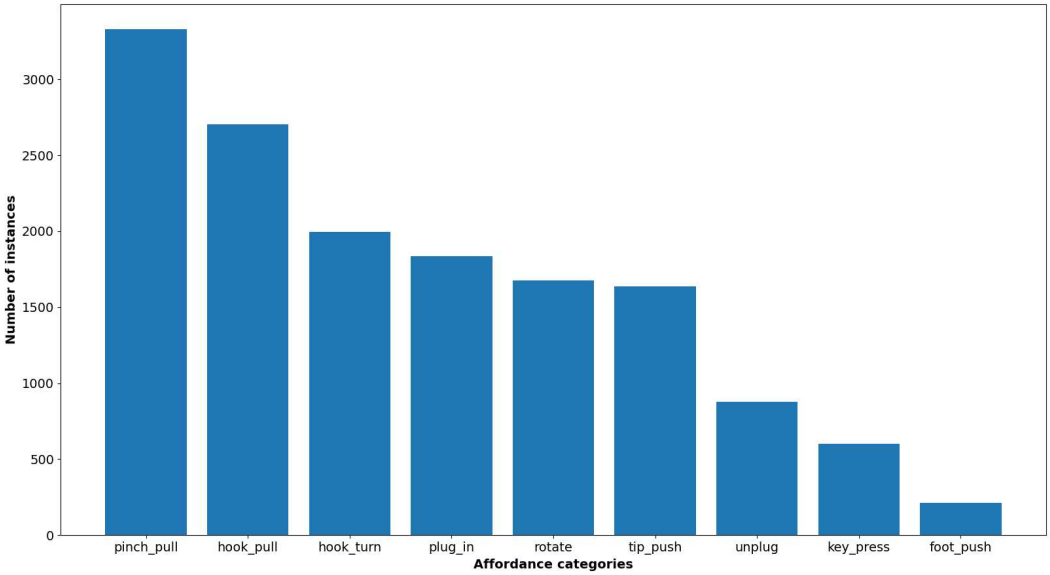

Figure 6. Distribution of affordance categories in the SceneFun3D dataset.

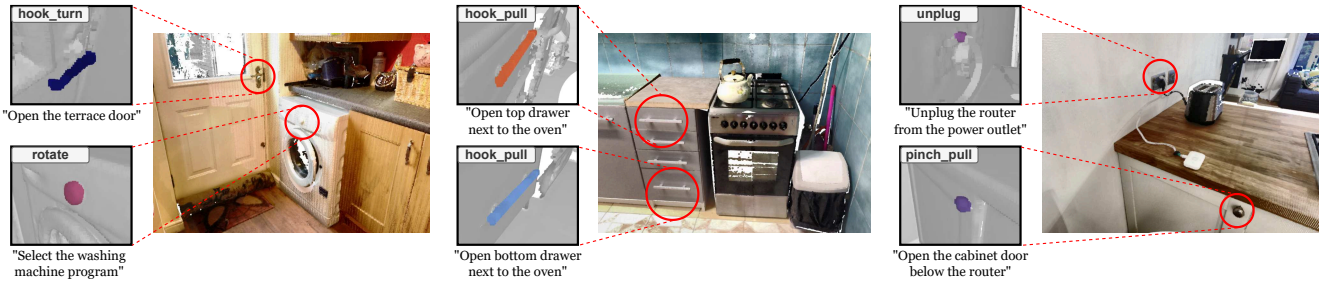

Figure 7. Additional examples on functionality annotations and natural language task descriptions.

SoftGroup-F. We also provide further details on our experimental setup on the SoftGroup-F baseline. We train our model for 1000 epochs using the Adam [3] optimizer and a cosine annealing schedule, with an initial learning rate of 0.002. The batch size is set to 2. We set the voxel size and grouping bandwidth to 2 cm and 4 cm, respectively. The backbone network is trained using the combined loss $\mathcal{L}_{\text{backbone}} = \lambda_{\text{m-dice}}\mathcal{L}_{\text{m-dice}} + \lambda_{\text{offset}}\mathcal{L}_{\text{offset}}$, where we experimentally set $\lambda_{\text{m-dice}} = 2$ and $\lambda_{\text{offset}} = 1$. We train the whole network using the multi-mask loss $\mathcal{L}_{\text{top-down}} = \lambda_{\text{ce}}\mathcal{L}_{\text{ce}} + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}} + \lambda_{\text{score}}\mathcal{L}_{\text{score}}$, where $\lambda_{\text{ce}} = 1$, $\lambda_{\text{dice}} = 2$ and $\lambda_{\text{score}} = 1$. We also use curriculum masking with an initial expansion radius $r_0 = 0.1$, a decay rate $d = 0.5$ and a decay interval $\alpha = 200$.

LERF. We rely on the original implementation of LERF based on NerfStudio [9]. We adapt the dataset loader for ARKitScenes and optimize a language-embedded neural implicit scene representation for each scene separately, using the default learning parameters of LERF. Once a scene representation is optimized, we can query it with arbitrary text queries. Since the language features originate from large pre-trained visual language models (VLM), i.e., CLIP, we do not require supervised learning. During inference, we can directly query the scene with the CLIP-text-encoding of the functional representations. The text prompts are performed on a per-frame basis which yields a relevancy field. This field is computed for each training frame and projected onto the provided 3D point clouds for evaluation. The final per-point relevancy scores are averaged over all views.

3.2. Models for task-driven affordance grounding

OpenMask3D-F. OpenMask3D [8] is a 3D open-vocabulary instance segmentation method. Given a 3D reconstructed geometry and posed RGB-D images, OpenMask3D obtains a per-mask features representation of the objects in the scene. In order to run OpenMask3D and our OpenMask3D-F on scenes from the ARKitScenes [1] dataset, we use the cropped Faro laser scans (which were obtained as described in Sec. 2.1) as the 3D geometry, and the low-resolution RGB-D images from the wide camera,

with resolution 256×192. We process 1 frame per each 60 frames. For obtaining OpenMask3D mask proposals using the original Mask3D backbone, we use 100 queries, whereas for OpenMask3D-F we use the masks predicted by Mask3D-F, where we used 80 queries. Some sequences in ARKitScenes were captured with varying device orientations, resulting in images that are rotated with respect to the ground plane. For such sequences, we ensure that each image is naturally oriented (which would improve the quality of the CLIP features) by rotating the RGB-D images by 90°, 180° or 270° depending on the camera orientation, prior to running OpenMask3D feature extraction module. Once we use description queries to retrieve object instances, we accept or reject proposals if the OpenMask3D embedding and text CLIP embedding has a cosine similarity score higher than 0.2 (note that this score does *not* correspond to an *absolute score* for confidence) and was set empirically.

LERF. LERF [2] embeds language features into a radiance field. For our experiments, we rely on the official implementation using NerfStudio [9] and optimize a NeRF representation with language-embedded features. Similar to our experiments with OpenMask3D, we use the low-resolution RGB-D images from the wide camera, with resolution 256×192. To be consistent with our OpenMask3D experimental procedure, we use 1 frame per each 60 frames for our LERF experiments.

Evaluation. For evaluating task-driven affordance grounding, we perform the following steps: For each scene, there are one or more ground truth task description annotations, such as "Open the bottom drawer of the wardrobe" and "Turn on the ceiling light". After obtaining an open-vocabulary scene representation using the previously described methods, we embed the annotated task descriptions using a CLIP [6] text encoder, and for each scene, we retrieve scene parts that are closest to the text-embedding of the descriptions annotated for that scene. We evaluate the task-driven affordance grounding performance by computing AP, AP_{50} and AP_{25} metrics.

3.3. Models for 3D motion estimation

Mask3D-FM. As described in the main paper, we add three linear projection heads to the Mask3D-F pipeline, which use as input the query features to predict the motion parameters, i.e., motion type, motion axis and motion origin. We parameterize the motion axis and motion origin as 3-dim vectors. We use the weights of the trained Mask3D-F model for initialization and further refine them for the task of motion estimation. We use the same hyperparameters as in Mask3D-F for training. Our loss is formulated as $\mathcal{L} = \mathcal{L}_{\text{seg}} + \mathcal{L}_{\text{motion}}$, where $\mathcal{L}_{\text{motion}}$ is defined as $\mathcal{L}_{\text{motion}} = \lambda_{\text{type}}\mathcal{L}_{\text{type}} + \lambda_{\text{axis}}\mathcal{L}_{\text{axis}} + \lambda_{\text{origin}}\mathcal{L}_{\text{origin}}$. We empirically set $\lambda_{\text{motion}} = 2$, $\lambda_{\text{axis}} = 14$ and $\lambda_{\text{origin}} = 14$.

4. Qualitative results on LERF

We also present qualitative results using the LERF model. In Fig. 8 we visualize the response field of LERF on a training frame given a text query.

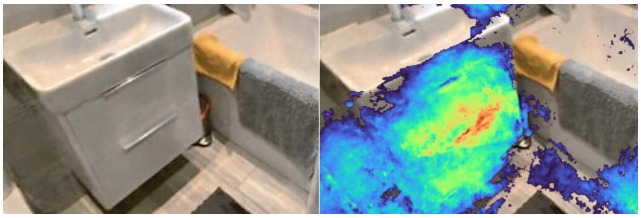

Figure 8. **Response field of LERF [2].** Query is "Open the bottom drawer." Red means high response, blue low response.

5. Applications

Our work encourages research on a spectrum of potential applications, including robotics and augmented reality.

5.1. Rendering virtual humans and avatars

Generating diverse and realistic human-scene interactions is crucial in AR applications. Current approaches on this task search for contact positions on the object surface in the 3D scene to synthesize physically possible interactions. By providing accurate 3D affordance masks in high-resolution 3D scenes, we facilitate the generation of realistic human grasps and human-object contact areas. In Fig. 9, we show examples of humans interacting with the functional elements in scenes of our dataset.

5.2. Robotics

Having the ability to understand and localize visual affordances is vital for robots to operate in dynamic and complex environments. In many domains, such as assistive robotics, humans often use natural language to command

Figure 9. Virtual human-scene interactions.

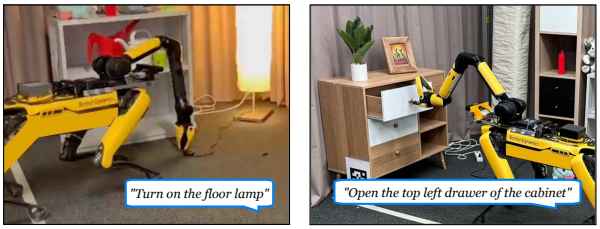

Figure 10. Robot-scene interaction. The ability to detect functional elements, enables a robot to perform scene interactions such as turning on a light, or opening a drawer.

an embodied agent to accomplish complex tasks in a 3D indoor environment. By building a large-scale dataset with accurate masks of functional interactive elements and natural language descriptions of tasks that involve interacting with them, we aid robotic agents in recognizing suitable areas for manipulation in the scene and following instructions expressed in natural language. In Fig. 10 we show examples of a robotic agent interacting with scene elements to achieve task-specific goals.

6. Limitations

Despite the large scale of our dataset, it might exhibit geographic biases. The design and 3D shape of the functional interactive elements present differences among different countries and continents. Therefore, to increase the diversity of our dataset and build more robust scene understanding methods it would be beneficial to capture additional 3D environments across the world. Furthermore, our dataset comprises static scenes. Thus, it does not contain information on the rigid movement and articulation of objects. Extending our pipeline to incorporate high-resolution laser scans in multiple states will aid in more detailed and realistic modeling of the interactions with the functional elements.

References

[1] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. ARKitScenes: A Diverse Real-world Dataset for 3D Indoor Scene Understanding Using Mobile RGB-D Data. *International Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 5

[2] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. LERF: Language Embedded Radiance Fields. In *International Conference on Computer Vision (ICCV)*, 2023. 6

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. 4

[5] Daniel Meister, Shinji Ogaki, Carsten Benthin, Michael Doyle, Michael Guthe, and Jiri Bittner. A Survey on Bounding Volume Hierarchies for Ray Tracing. *Computer Graphics Forum*, 2021. 1

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*, 2021. 6

[7] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. In *International Conference on Robotics and Automation (ICRA)*, 2023. 4

[8] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. OpenMask3D: Open-Vocabulary 3D Instance Segmentation. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2023. 5

[9] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH*, 2023. 5, 6