2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# SAGrid: Scaling Robot Simulation through Automatic Affordance Annotation on In-the-Wild 3D Assets

Cem Gokmen*†, Yalcin Tur*, Aditesh Kumar, Auddithio Nag and Li Fei-Fei

***Abstract*— Robot simulation is a highly efficient approach for scaling data collection for robot learning, but scaling for most household tasks remains bottlenecked by a shortage of simulation-ready 3D assets. While modern robot simulators can model complex phenomena like temperature and fluids, most in-the-wild 3D models lack "simulation affordances" (specialized annotations such as fluid source and heat emitter positions) that are required for these features. As a result, costly manual annotation is required, severely limiting asset scale and variety.**

**We introduce Simulation Affordance Grids (SAGrid), a method that automates the annotation of simulation affordances on in-the-wild 3D meshes. SAGrid leverages pretrained representations (DINOv2, TRELLIS) to predict a dense 3D distance field to the nearest affordance. Our approach operates effectively in a low-data regime, requiring as few as 10 training objects per affordance type to accurately locate these features. We validate our method by processing Objaverse-XL models and integrating them into the BEHAVIOR-1K simulator. Training robot policies on this automatically expanded asset suite significantly improves generalization to unseen objects in complex tasks, demonstrating that automated affordance annotation is crucial for scaling robot learning.**

## I. INTRODUCTION

Robot simulation has been established in recent years as an efficient source of data for training robot learning algorithms. It effectively addresses some of the most critical issues that limit robot learning today: in imitation learning, it is a cheap source of successful demonstrations that are otherwise very expensive to obtain from human experts, and, in reinforcement learning, it is a highly scalable way of performing rollouts with dense reward signals.

For much of its history, robot simulation has been limited to rigid body dynamics, where a world entirely consisting of some fixed and some movable rigid bodies is simulated from physics first principles. While rigid body dynamics have been a useful playground for testing robot learning algorithms on pick-and-place tasks, more is needed to simulate the complex tasks required for practical, real-world applications. A typical household task like cleaning, for example, involves different physical phenomena such as interacting with fluids and soft bodies. Other tasks like cooking involve changes to the object's nature through freezing, cooking, burning, melting, and so on.

The simulation task is further complicated by the need to correctly model how these phenomena are introduced into the world, e.g. water flowing out of a sink or heat radiating out of a stove. These devices, in real life, work through mechanisms like indoor plumbing that are neither easy nor necessary to simulate inside a robot simulator. Instead, they are typically simulated via special annotations on the 3D mesh that indicate to the simulator where these phenomena such as water particles or heat should be produced. We refer to these annotations as **simulation affordances**. For most object datasets that support these features ([1], [2], [3]), these affordances were manually annotated by human experts, as a result, even though they provide for very high-quality feature demonstrations for those platforms, their scale is limited to tens of objects.

Fig. 1. Simulation Affordances are needed for simulating a variety of features that are necessary for household robotics. Left: toggle button and heat source affordances are used to simulate a stove. Center: toggle button and fluid source affordances are used to simulate a faucet. Right: toggle button and particle remover affordances are used to simulate a vacuum cleaner.

The problem with this method of affordance annotation is that, while robot simulation does, in theory, have the advantage of infinite scalability with compute, in practice the value of this scalability is limited by the lack of diversity in robot simulation scenes and assets. For example, training a policy while randomizing object and robot configurations around one particular 3D mesh in one scene will not make the learned policy generalize to more objects in more scenes. As a result, the true potential of robot simulation can only be realized if the diversity of scenes and assets can be scaled along with the number of robot trajectories. Although valuable sources of 3D mesh data exist, such as large-scale datasets and generative models, they do not provide an easy avenue for obtaining simulation affordance annotations. As a result, neither kind of data can be used easily to scale up simulation and data collection on tasks beyond simple navigation and pick-and-place.

In this work, we address this shortcoming by introducing *Simulation Affordance Grids (SAGrid)*, a method for automatically annotating the necessary simulation affordances on in-the-wild 3D meshes, thus allowing them to be used for scaling robot learning without incurring the cost of manual annotations. SAGrid takes a pretrained sparse feature

*Equal contribution
†Corresponding author: cem@cemgokmen.com
All authors are with the Department of Computer Science, Stanford University, Stanford, CA, USA.

grid representation from TRELLIS [4] and an object-level DINOv2 [5] feature as input, and predicts the distance of the nearest simulation affordance from each point on the sparse grid. A separate instance of SAGrid is trained for each kind of simulation affordance (e.g. fluid source, toggle button, heat source). Individual affordances can then be positioned through clustering or multilateration. SAGrid can be trained in regimes with very few existing annotated models (experimentally, as low as 10) to predict relevant simulation affordances, and then be used to significantly expand, without the use of human annotator labor, the set of simulation-ready task assets. In our experiments, we convert in-the-wild stove and sink objects from the Objaverse-XL [6] dataset into the BEHAVIOR-1K [1] simulation environment, we show that SAGrid successfully predicts the correct affordance annotations for these objects, and we prove that this additional data diversity significantly improves policy generalization on unseen objects.

## II. RELATED WORK

A rich ecosystem of robot simulators has been developed over the years to facilitate the training of embodied agents. Early simulators like Gibson Env [7] and Habitat [2] focused on navigation in entirely static scenes. The next generation of simulators like iGibson [8] and Habitat 2 [9] supported rigid-body dynamics with some floating or articulated objects. Later generations like iGibson 2 [10] introduced some of the simulation behaviors discussed here, like fluid and heat simulation reliant on affordance annotations. Others like AI2Thor [11] also supported affordance-based simulation of certain features while forgoing rigid body physics. BEHAVIOR-1K [1] greatly expanded the set of such simulation features by supporting affordances like spray cones, vacuum cleaners, sharp edges on cutting tools, and so on. More recent works like RoboCasa [3] have focused on small sets of high-quality affordance-annotated objects that support features like knob-burner correspondences on stoves. However, a common limitation of these more advanced simulators is the scale of their asset libraries; they typically ship with hundreds to a few thousand objects across all categories combined, which restricts the diversity available for training generalizable policies for any specific tasks on any narrow set of objects.

To address the need for asset diversity, the community has curated large-scale datasets of 3D meshes, such as ShapeNet [12], Objaverse [13], and Objaverse-XL [6]. These repositories contain millions of "in-the-wild" 3D models, offering immense variety in geometry and appearance. Complementing these datasets are powerful generative models like TRELLIS [4] and Hunyuan-3D [14], which can synthesize novel 3D assets based on image or text prompts. The primary advantage of these sources is their scale, which is crucial for training policies that generalize. However, a significant drawback is that these assets lack the specific physical and semantic annotations required for complex interaction simulation. They are typically provided as raw meshes, and are not immediately usable in robot simulators without significant manual effort to add affordances for features like fluid sources, heat sources, or buttons.

Recent advancements in self-supervised learning have produced powerful pretrained visual representations that can be beneficial for the pursuit of simulation affordances. Beginning with models like CLIP [15], semantically meaningful pretrained embedding spaces became accessible. Models like DINO [16] and DINOv2 [5] provide rich semantic features that are useful for a wide range of downstream tasks. These powerful 2D feature extractors have been foundational for new methods that interpret 3D space. Works like LERF [17] and D3Fields [18] unproject these 2D embeddings back into 3D, creating semantic fields that can be queried to understand object parts, affordances, or relationships within a scene. Our work builds on this principle, leveraging pretrained 2D features to imbue a 3D grid with the semantic understanding needed to locate simulation-specific affordances.

Our work is also related to articulation prediction. Prior articulation methods infer movable joints from robot interaction or reconstruction cues [19], [20], but they do not address function-specific affordances on unlabeled in-the-wild meshes, which is the setting of our work.

## III. SAGRID

We study *simulation affordance prediction* on 3D meshes: given a textured 3D mesh, the goal is to predict affordance annotations $A = \{a_1, \ldots, a_n\}$, where each $a_i = (p_i, c_i)$ contains a 3D position $p_i$ and an affordance label $c_i \in C$ such as `togglebutton`, `heatsource`, or `fluidsource`. The affordance positions should simulate the real-life behavior of the represented 3D object, for example, a faucet has water coming out of the tip, and a stove produces fire above the heating elements. The full inference pipeline must also run autonomously on unlabeled mesh collections. SAGrid addresses this task by learning a distance-to-nearest-affordance field over a voxelized object representation, with separate models for each affordance type.

### A. Architecture

The core of our system is a 3D CNN operating on a dense $64^3$ voxel grid derived from the TRELLIS SLAT representation [4]. The input combines:

- **Geometric Features**: An 8-channel-per-voxel representation capturing local geometric properties of the object from a pretrained TRELLIS [4] SLAT (Structured Latents) encoder model ('TRELLIS-image-large'). Pretrained as part of an autoencoder that can reconstruct a textured mesh, this representation is rich in terms of the local geometry and texture at a given voxel.
- **Semantic / Visual Features**: A 1024-channel-per-object feature vector from a pretrained DINOv2 [5] model ('dinov2_vitl14_reg'), providing rich semantic information aggregated from 150 rendered views of the object. This global feature vector is concatenated with the features of every voxel, providing the convolutional network with object-level context that complements its local receptive field.

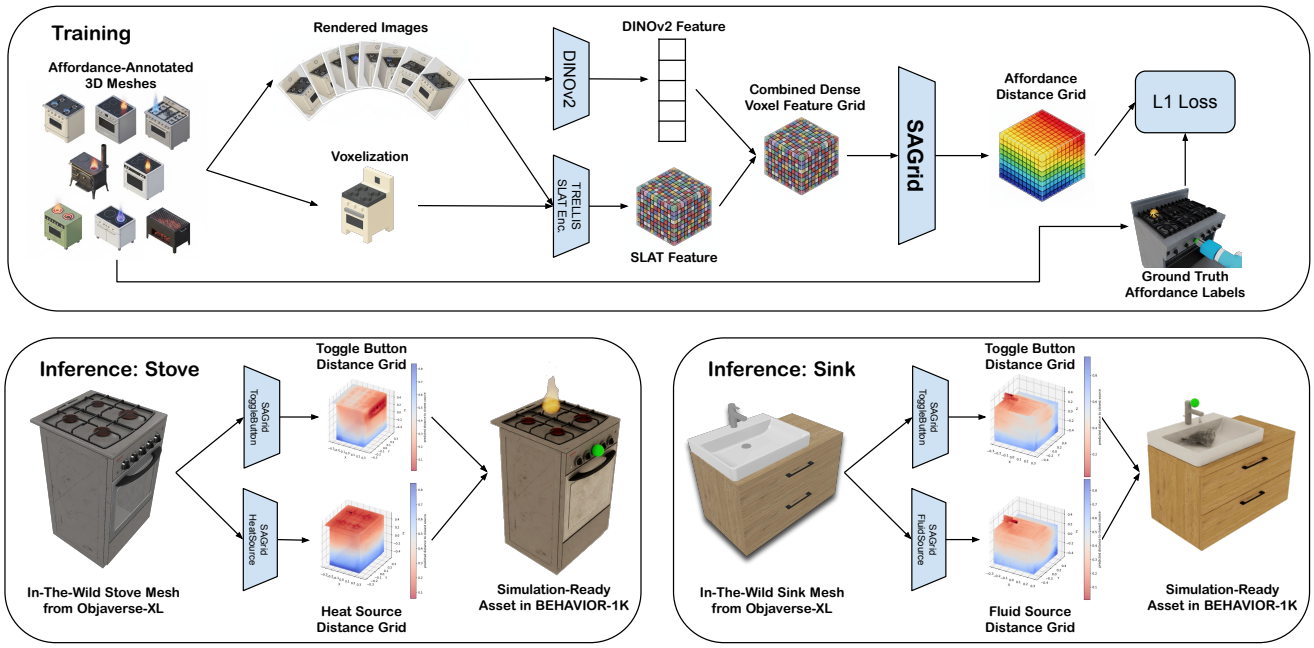


Fig. 2. The SAGrid training and inference processes. At training time, objects are preprocessed through the feature extraction pipeline, and the model is trained for the task of predicting the distance from each grid point to the nearest simulation affordance. At inference time, in-the-wild objects without any labels are labeled automatically with the correct simulation affordances.

- **Voxel Mask**: A single-channel binary mask indicating which voxels are actually occupied by the object.

This results in an input tensor with 1033 channels for each voxel. The network architecture consists of a stack of 3D convolutional layers (exact architecture discussed in Experiments), each followed by batch normalization and a ReLU activation function. The final prediction head uses a $1 \times 1 \times 1$ convolution and a Softplus activation to output a single-channel distance grid of size $64^3$.

### B. Feature Extraction

Both during training and inference, we process objects through a feature extraction pipeline to obtain the input multimodal voxel feature grid. We build upon the steps in the TRELLIS preprocessing pipeline:

1) We render the object from 150 random viewpoints.
2) We compute patch-level features by feeding the renderings through the pretrained DINOv2 model.
3) We convert the object into a sparse voxel grid representation (normalizing the object into a $[-0.5, 0.5]^3$ unit bounding box).
4) We compute the structural latents representation by feeding DINOv2 patch features and the sparse voxel grid representation through the pretrained TRELLIS encoder.
5) We unify the DINOv2, sparse voxel grid, and structural latents into a single dense voxel feature grid representation.

### C. Training Process

We train the network to predict the distance from each voxel to the nearest ground-truth affordance point $c^*$. To smooth the optimization landscape, we regress the logarithmic distance and minimize an L1 loss over occupied voxels:

$$D(v) = \log(1 + ||v - c^*||_2)$$

$$\mathscr{L} = \frac{1}{|V_{mask}|} \sum_{v \in V_{mask}} |D_{pred}(v) - D_{gt}(v)|$$

To improve generalization, we employ a comprehensive 3D data augmentation pipeline during training, including random translations, 90-degree rotations around the Z-axis, reflections across the X and Y axes, and warping. We use the AdamW optimizer with a cosine learning rate scheduler. A separate model is trained for each affordance type (e.g., 'toggleglebutton', 'fluidsource', 'heatsource') to create specialized distance fields. We also record object-level DINOv2 feature averages for object retrieval during inference.

### D. Inference Process

At inference time, we first filter a large unlabeled mesh collection with cosine similarity in DINOv2 feature space, then run the full feature pipeline and predict a dense distance grid for each candidate object. We additionally generate labels of object masses with VLMs [21], [22] and collision geometry with CoACD [23].

When predicting one affordance instance, we choose the occupied voxel with minimum predicted distance:

$$p_{pred} = \underset{v \in V_{mask}}{\operatorname{argmin}} \; D_{pred}(v)$$

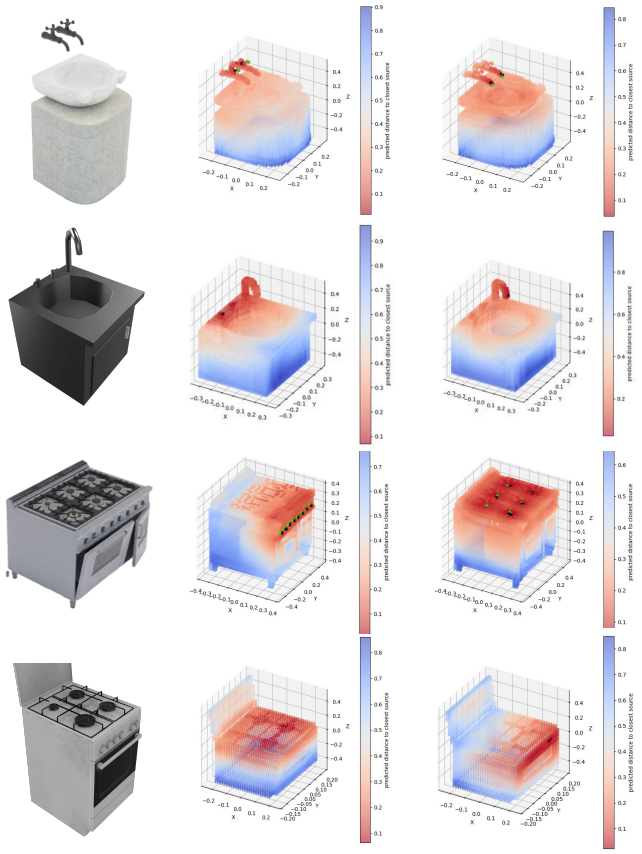

Fig. 3. Representative affordance predictions on stove and sink assets from BEHAVIOR-1K and Objaverse-XL (left: togglebutton, right: heat/fluid source). Ground-truth affordance locations for BEHAVIOR-1K are shown with green dots.

For multiple known instances, we threshold low-distance voxels and cluster them with k-means.

Predicted points are then mapped back to the original object coordinates and projected onto the collision mesh surface.

## IV. EXPERIMENTS

We designed our experiments to evaluate two key aspects of SAGrid: 1) the accuracy of its affordance predictions and 2) the utility of these automatically generated annotations for improving the generalization of downstream robot learning tasks.

We use the BEHAVIOR-1K [1] assets and simulator both as the training data for our networks and as the simulation environment to simulate the affordances, collect demonstrations, and evaluate policies. For our robot learning experiments, we focus on two tasks: cooking a steak on a stove, and filling a cup with water from a faucet. These tasks were selected because they constitute skills that are necessary to succeed in many of the 1,000 BEHAVIOR-1K tasks. They are also both reliant on non-rigid-body phenomena.

We perform all of our experiments on two categories of objects that are involved in the above tasks: sinks (with faucets included) and stoves. We train a total of four SAGrid networks: {stove_heatsource, stove_togglebutton, sink_fluidsource, sink_togglebutton}. Note that we train separate toggle button predictors for the two different kinds of objects to simplify the task and improve the performance of our small CNNs. This task selection naturally places us in a low-data regime since the dataset only contains 31 sinks and 14 stoves that are fully annotated with simulation affordances. We set apart 5 sinks and 5 stoves as a validation set of objects: these objects are not included either in the SAGrid network training or the policy network training, and are used both for affordance prediction evaluation, and for policy evaluations. We use the rest of the objects as our training dataset.

| Model ID | CNN Architecture | Error (cm) |
|---|---|---|
| CNN - 1 | [3,1,1024],[3,4,512,4],[3,1,256],[3,1,128] | 4.84 |
| CNN - 2 | [3,1,32],[3,1,16] | 6.59 |
| CNN - 3 | [3,1,128],[3,4,64,4],[3,1,32],[3,1,16] | 6.78 |
| PointNet | Point cloud architecture | 7.13 |
| CNN - 4 | [3,1,256],[3,4,128,4],[3,1,64],[3,1,32] | 7.25 |
| CNN - 5 | [3,2,32,2],[3,1,16] | 10.29 |
| CNN - 6 | [3,1,32],[3,2,16,2] | 11.47 |
| MLP | [512], [256], [128] | 12.87 |

Table I. Architecture comparison. For CNN layers: [kernel, padding, out_channels, dilation]; for MLP: [layer sizes]. Dilation defaults to 1 if omitted.

We use the Objaverse-XL [6] dataset's Sketchfab subset as our in-the-wild 3D mesh dataset. Objects from this dataset were fed through the SAGrid pipeline to obtain more sinks and stoves with the appropriate simulation affordances. These objects were then imported into the BEHAVIOR-1K simulator.

### A. Affordance Prediction Performance by Architecture

In this experiment, we compared several low-data architectures for the SAGrid prediction head, including MLP, PointNet [24], and 3D CNN variants. The best model was a large-receptive-field 3D CNN, which achieved 4.84 cm mean localization error on the validation set and outperformed PointNet (7.13 cm), a smaller CNN (6.59 cm), and an MLP (12.87 cm). We use this model in all remaining experiments; much of its residual error lies along the surface normal and is reduced by the final mesh-projection step.

The large-receptive-field CNN-1 architecture significantly outperformed the alternative architectures and was selected as the default SAGrid architecture for the rest of the experiments. Qualitatively, we observe that the 4.8 cm error stems mainly from distance between the affordance and the object surface, which we remedy in our final inference pipeline by projecting the affordance point onto the object's collision mesh.

We also quantitatively examine the output of our selected SAGrid model on a sample of the validation set objects in Figure 3. Predictions are semantically consistent across both sources and the predicted minima align well with ground-truth affordance regions.

### B. Comparison with Alternative Affordance Prediction Approaches

To contextualize SAGrid's performance, we compare it against three alternative paradigms for simulation affordance prediction on the same held-out validation set, measured by mean localization error (meters; lower is better).

**Embedding Search** averages DINOv2+TRELLIS embeddings at the known affordance positions from the training objects. At inference time, it computes similarity between the inference object's embedding field and the known average affordance embedding from the training set, and predicts the position of the highest-similarity embedding as the affordance position. This simple approach achieves 0.1654 m average error, which shows that the embedding field contains some relevant information, even in the absence of the purpose-built SAGrid distance classifier.

**VLM Triangulation** prompts a vision-language model (Gemini 2.5 Flash [22] or Qwen3-VL [25]) with multi-view renderings and asks it to identify the 2D pixel location of each affordance per view, then computes a 3D position prediction using one of the two methods listed below.

- *ray-ransac* converts the 2D pixel to a 3D ray from the camera, iteratively selects pairs of rays and selects the solution with the most inlier views (where a view is counted as an "inlier" if its ray crosses within some fixed distance of the three rays' intersection), making it robust to outlier predictions from individual viewpoints.
- *point-ransac* converts the 2D pixels to 3D points by performing ray-casting from the camera. It then iterates over the predicted 3D points and selects the solution with the most neighboring points.

The best result (Gemini 2.5 Flash, ray_ransac, 0.1285 m) shows VLMs carry meaningful spatial priors, but errors remain significantly larger than SAGrid.

**Image Generation** uses Google NanoBanana 2.5 to augment the rendered viewpoints with markers over the affordance positions. It then back-projects the predicted 2D location into 3D. The position is found using the same methods outlined in the VLM triangulation section. This approach achieves a minimum average error of 0.2311 m, performing significantly worse than SAGrid.

As shown in Table II, SAGrid averages 4.8 cm in error and outperforms all baselines by a large margin across every affordance type, validating the benefit of learning a dedicated 3D distance field over retrieval- or generation-based alternatives.

| Model | Method | toggle sink | fluid source | toggle stove | heat source | Avg |
|---|---|---|---|---|---|---|
| *Embedding Search* | | | | | | |
| DINOv2+TRELLIS | cosine | 0.0766 | 0.1012 | 0.1959 | 0.2881 | 0.1654 |
| DINOv2+TRELLIS | L2 | 0.1013 | 0.1046 | 0.2100 | 0.2607 | 0.1691 |
| *VLM Triangulation* | | | | | | |
| Gemini 2.5 Flash | ray-ransac | 0.1057 | 0.1192 | 0.1428 | 0.1462 | 0.1285 |
| Gemini 2.5 Flash | point-ransac | 0.1354 | 0.1621 | 0.1959 | 0.1731 | 0.1666 |
| Qwen3-VL 8B | ray-ransac | 0.2196 | 0.2559 | 0.2070 | 0.2613 | 0.2360 |
| Qwen3-VL 8B | point-ransac | 0.4391 | 0.5067 | 0.1295 | 0.2695 | 0.3362 |
| *Image Generation* | | | | | | |
| NanoBanana 2.5 | ray-ransac | 0.2219 | 0.2040 | 0.2838 | 0.2147 | 0.2311 |
| NanoBanana 2.5 | point-ransac | 0.3788 | 0.3818 | 0.3042 | 0.1607 | 0.3064 |
| *SAGrid (Ours)* | | | | | | |
| CNN-1 | distance field | **0.038** | **0.033** | **0.057** | **0.064** | **0.048** |

Table II. Mean localization error (m) ↓ across affordance prediction paradigms. SAGrid outperforms all baselines by a large margin.

Fig. 4. The steps of a steak cooking task. The robot must approach the stove (left), toggle on the stove by pressing the button (center), place the pan on the heat source affordance (right), and wait for the steak to cook.

### C. Robot Learning Experiments

For each task below, we train 25 Diffusion Policy [26] networks via the LeRobot [27] framework, varying the number of unique objects in $\{1, 5, 10, 25, N_{\max}\}$ and training trajectories in $\{50, 100, 500, 1000, 2000\}$, with trajectories distributed evenly across objects. The first 10 objects are original BEHAVIOR-1K assets; additional objects are Objaverse-XL meshes annotated by SAGrid. Each episode begins with the target object already grasped to abstract away grasping. Demonstrations are generated via heuristic waypoints and a motion planner from randomized starting configurations. All policies are trained for 10,000 steps (4 observation steps, 12 action steps, 24-step horizon) and evaluated on 100 rollouts with randomized configurations on held-out BEHAVIOR-1K objects.

*1) Cooking Steak:* In this task, the robot must activate a stove burner via its toggle button (`stove_togglebutton`) and place a pan with a steak on the heat source (`stove_heatsource`); the steak cooks after roughly two seconds of exposure (Figure 4). We train with up to $N_{\max} = 44$ stove objects.

As shown in Figure 5, object diversity is the dominant factor: the best success rate (49%) is achieved with 44 objects and 2000 trajectories, versus 31% when scaling only trajectories on a limited object set. Training on a single object saturates near 16% regardless of trajectory count, while spreading too few trajectories across many objects (e.g., 50 across 44) also fails (3%) due to insufficient per-object coverage.

*2) Cup Filling:* In this more challenging task, a dual-arm robot must turn on a faucet (`sink_togglebutton`) and position a cup beneath the water spout (`sink_fluidsource`) to fill it (Figure 6). This requires finer manipulation than the steak task. We train with up to $N_{\max} = 33$ sink objects.

Object diversity is again critical (Figure 7): training on a single sink remains near failure even with 2000 trajectories, while the best result (24%) is reached with 25 objects and 2000 trajectories. Performance drops slightly at 33 objects,

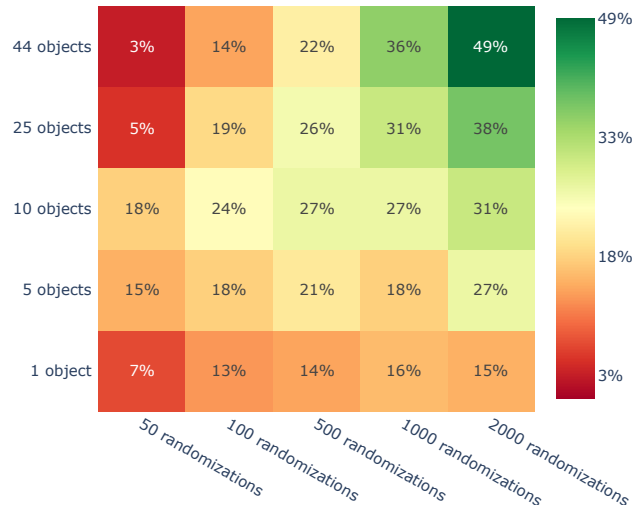


Fig. 5. Steak cooking policy success rate on 100 rollouts on held-out validation objects, by number of training trajectories and unique object models. The first 10 objects are from BEHAVIOR-1K; additional objects (up to 44) are Objaverse-XL assets annotated by SAGrid.

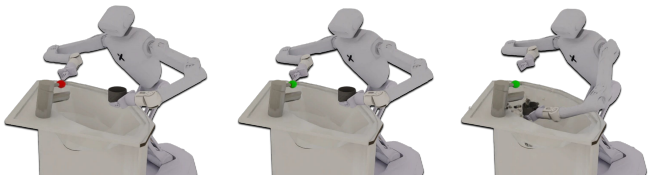

Fig. 6. The steps of a cup filling task. The robot must approach the sink (left), toggle on the faucet by touching the handle (center), place the cup under the fluid source affordance (right), and wait for the cup to fill up.

suggesting a trade-off between diversity and annotation noise in this harder task.

## V. LIMITATIONS AND FUTURE WORK

Our approach in its current form has a few important limitations.

The first limitation is that our current approach is only able to predict dimensionless and orientationless affordances, e.g., points in $\mathbb{R}^3$. However, many simulation affordances, such as the spray cone of an atomizer, the fillable volume of a cup, or the opening of a vacuum cleaner, are not dimensionless points but instead primitive shapes with parameters and a SE(3) pose. While it is possible to extend our current architecture to predict these, we found it difficult to do so satisfactorily in the current low-data regime. As a result, our model can only be used to generate point annotations such as toggle buttons, heat sources, and fluid sources. We would like to explore alternative approaches in these domains in the future.

Another limitation is that in our experiments, we currently only predict one instance per affordance type per object. This is *not* an intrinsic limitation of our approach: in fact, the distance-to-nearest-affordance output behaves as expected in the presence of multiple affordances, showing small distances around each affordance. However, extracting the affordance positions correctly from the distance grid is significantly more difficult when the number of affordances is unknown, requiring the use of an affordance-counting classifier. Since multiple affordances of the same type are already not supported by the BEHAVIOR-1K simulator, we chose to avoid this complexity and pick only one of the predicted simulation affordances. We plan to integrate an open-source image classifier as a follow-up to remedy this limitation, likely in the shape of a pre-trained vision-language model (VLM).

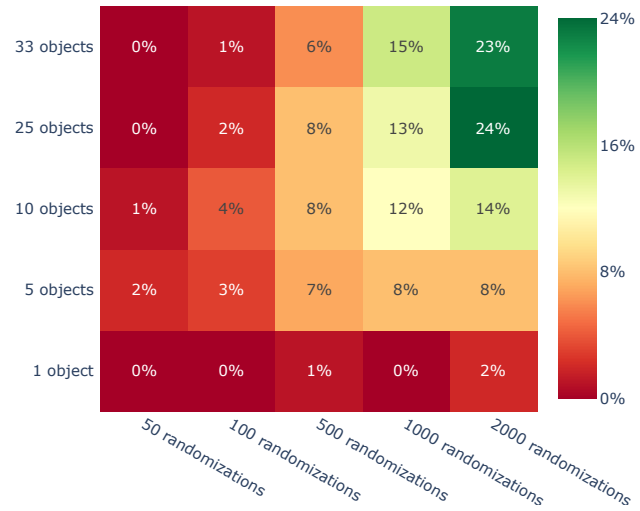


Fig. 7. Cup filling policy success rate on 100 rollouts on held-out validation objects, by number of training trajectories and unique object models. The first 10 objects are from BEHAVIOR-1K; additional objects (up to 33) are Objaverse-XL assets annotated by SAGrid.

## VI. CONCLUSION

In this paper, we addressed the critical challenge of asset scarcity in advanced robot simulation. We introduced Simulation Affordance Grids (SAGrid), a novel method that automatically predicts simulation affordances on unlabeled, in-the-wild 3D meshes by leveraging powerful, pretrained 2D and 3D feature representations. Our experiments demonstrated that SAGrid can be trained on a very small dataset of annotated objects to accurately identify affordances, such as heat sources on stoves and fluid sources on faucets, and we showed that policies trained on an augmented dataset exhibited significantly improved generalization on downstream robotics tasks. These results suggest that automated affordance annotation is a practical path toward scaling simulation asset diversity for robot learning.

## VII. ACKNOWLEDGEMENTS

The artwork in Figure 2 was generated with the assistance of AI models (e.g., Google Gemini 2.5 Pro).

This work was supported in part by the AI2050 program at Schmidt Sciences (Grant G-23-65947).

## REFERENCES


[1] C. Li et al., "Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation," *arXiv preprint arXiv:2403.09227*, 2024.

[2] Manolis Savva* et al., "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[3] S. Nasiriany et al., "Robocasa: Large-scale simulation of everyday tasks for generalist robots," *ArXiv*, vol. abs/2406.02523, 2024.

[4] J. Xiang et al., "Structured 3d latents for scalable and versatile 3d generation," *arXiv preprint arXiv:2412.01506*, 2024.

[5] M. Oquab et al., "Dinov2: Learning robust visual features without supervision," *ArXiv*, vol. abs/2304.07193, 2023.

[6] M. Deitke et al., "Objaverse-xl: A universe of 10m+ 3d objects," *ArXiv*, vol. abs/2307.05663, 2023.

[7] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079. DOI: 10.1109/CVPR.2018.00945.

[8] B. Shen et al., "Igibson 1.0: A simulation environment for interactive tasks in large realistic scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7520–7527. DOI: 10.1109/IROS51168.2021.9636667.

[9] A. Szot et al., "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[10] C. Li et al., "Igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 455–465.

[11] "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2017.

[12] A. X. Chang et al., "Shapenet: An information-rich 3d model repository," *ArXiv*, vol. abs/1512.03012, 2015.

[13] M. Deitke et al., "Objaverse: A universe of annotated 3d objects," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13 142–13 153, 2022.

[14] X. Yang et al., "Hunyuan3d 1.0: A unified framework for text-to-3d and image-to-3d generation," 2024.

[15] A. Radford et al., "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, 2021.

[16] M. Caron et al., "Emerging properties in self-supervised vision transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[17] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19 672–19 682, 2023.

[18] Y. Wang et al., "D3fields: Dynamic 3d descriptor fields for zero-shot generalizable rearrangement," in *Conference on Robot Learning*, 2023.

[19] S. Y. Gadre, K. Ehsani, and S. Song, "Act the part: Learning interaction strategies for articulated object part discovery," *ICCV*, 2021.

[20] Z. Mandi, Y. Weng, D. Bauer, and S. Song, "Real2code: Reconstruct articulated objects via code generation," *ArXiv*, vol. abs/2406.08474, 2024.

[21] O. A. Hurst et al., "Gpt-4o system card," *ArXiv*, vol. abs/2410.21276, 2024.

[22] G. Comanici et al., *Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities*, 2025. arXiv: 2507.06261 [cs.CL].

[23] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, pp. 1–18, 2022.

[24] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2016.

[25] Q. Team, *Qwen3 technical report*, 2025. arXiv: 2505.09388 [cs.CL].

[26] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," *ArXiv*, vol. abs/2303.04137, 2023.

[27] R. Cadene et al., *Lerobot: State-of-the-art machine learning for real-world robotics in pytorch*, https://github.com/huggingface/lerobot, 2024.