Point Cloud Affordance Highlighter

Amirali Changizi
Politecnico di Torino
amirali.changizi@studenti.polito.it

Meelad Dashti
Politecnico di Torino
meelad.dashti@studenti.polito.it

Kimia Dorrani
Politecnico di Torino
kimia.dorrani@studenti.polito.it

Abstract

Detecting affordance regions in 3D data is a challenging task due to diverse geometries and representations like meshes and point clouds. While crucial for robotics and human-object interaction, existing methods rely on labeled datasets or geometric features, limiting their scalability. We utilize a novel label-free pipeline for affordance detection using neural fields and pre-trained vision-language models. Building on the 3D Highlighter architecture, our framework uses differentiable rendering and CLIP embeddings to semantically localize affordance regions on point clouds through text prompts. The method eliminates manual annotations while adapting seamlessly to various 3D formats. Results demonstrate acceptable performance for unsupervised object manipulation and interaction systems. Our code is available here. index terms– affordance regions, neural field, vision-language models, 3D Highlighter

1. Introduction

The advancement of robotics and intelligent systems demands sophisticated object interaction capabilities in real-world environments. A crucial aspect of this interaction is affordance detection– identifying regions of objects optimized for specific interactions like gripping, sitting, or pouring. While computer vision has made significant strides in object detection and segmentation, affordance detection presents unique challenges in bridging visual perception with functional understanding. Traditional affordance detection methods rely heavily on supervised learning with extensive annotated datasets or geometric approaches that struggle with complex, ambiguous functionalities. These limitations have motivated the exploration of alternative approaches. Recent advances in vision-language models like CLIP [6] and neural field-based representations [7] offer promising solutions by enabling semantic understanding and flexible 3D geometry representation. Building on the 3DHighlighter architecture [3], we present a system that combines neural fields, differentiable rendering [1], and pre-trained vision-language models for affordance detection in 3D data. Our framework projects 3D point clouds to 2D renders and uses CLIP to evaluate alignment between rendered views and text prompts through embedding similarity. This approach eliminates the need for manual annotations or 3D pre-training while maintaining adaptability across different 3D data formats.

2. Related Work

2.1. Traditional Affordance Detection

Traditional affordance detection relied on geometric feature extraction and manual annotation, utilizing handcrafted features like surface curvature and shape descriptors to identify interaction regions. Rule-based systems mapped these features to affordance categories - for example, identifying flat surfaces as ”support” regions or cylindrical shapes as ”grasp” regions. While effective for simple geometries, these methods struggled with complex interactions and required extensive manual annotation. These limitations ultimately motivated the development of more flexible learning-based approaches.

2.2. 3D Affordance Detection and Datasets

The development of 3D affordance understanding was significantly advanced by Deng et al.’s 3D AffordanceNet [4], which established protocols for full-shape, partial-view, and rotation-invariant affordance estimations. While the approach successfully demonstrated affordance learning from 3D data, it relied heavily on supervised learning with manual annotations. While their exploration of semi-supervised learning methods was promising, it emphasized the need for more efficient, label-free approaches - a limitation that motivates our work’s focus on leveraging pre-trained vision-language models to reduce dependency on manual annotations.

2.3. Interaction-driven 3D Affordance Grounding Network (IAG)

IAG learns affordances by aligning 2D interaction images with 3D representations [8]. However, its dependence on specific datasets and pre-defined categories motivated our more flexible vision-language approach to semantic localization.

3. Method

3.1. Original Pipeline Overview

The initial pipeline proposed in the original paper utilizes a 3D mesh as input to a neural network that aims to highlight probabilities for each mesh vertex. Subsequently, the 3D mesh is projected into 2D space through camera-based rendering techniques. The rendered image, along with a descriptive text prompt, is processed by the CLIP model to generate corresponding image and text embeddings. The cosine similarity between these embeddings is computed, and its negative value serves as the loss function for training the neural network.

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta) = -\frac{e_I \cdot e_T}{|e_I| \cdot |e_T|} \tag{1}$$

During experimental validation, our team encountered challenges in reproducing the results reported in the original paper, despite using the authors' provided implementation. As illustrated in Fig. 1 and Fig. 2 and Tab. 1, the output exhibits significant variability across different objects, camera angles, and hyperparameter configurations.
These findings suggest that achieving consistent affordance highlighting remains a challenging task without a definitive solution. Based on extensive experimentation involving modifications to the neural network architecture, optimization of training hyperparameters, and implementation of various camera rendering techniques, significant improvements remained elusive. Analysis of these experimental results revealed a fundamental limitation: the CLIP model's inherent difficulty in establishing precise spatial correspondences between localized image regions and their textual descriptions as well as being biased towards holistic object classification rather than part segmentation.

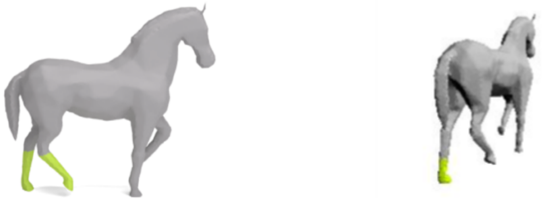


Figure 1. Left image is from the original paper. the right is the image generated by running the demo code released from the paper's researchersWe optimize 3D Highlighter on a mesh of a horse with the target localization 'shoes'.

Figure 2. Left image is from the original paper. the right is the image generated by running the demo code released from the paper's researchers.We optimize 3D Highlighter on a mesh of a dog with the target localization 'hat'

Hyperparameter	Value	Clip-score
Learning rate	0.001	-0.3425
Learning rate	0.0001	-0.3486
Learning rate	1e-05	-0.3352
Depth	3	-0.3200
Depth	4	-0.3376
Depth	5	0.3181
Number of views	3	-0.3408
Number of views	5	-0.3471
Number of views	7	-0.3181

Table 1. Exploring different hyperparameters for the prompt: A 3D render of a gray candle with highlighted hat

3.2. Adaptation to Point Cloud

We adapt the current pipeline to processing Point Cloud, which is a crucial data type, since it is more practical for real-world applications compared to meshes which often need to be manually created or reconstructed. Point clouds can be directly captured from various real-world sensors like RGB-D cameras, LiDAR, and other 3D scanning devices. Point clouds represent the raw 3D data as captured, preserving the original geometric information. Meshes require additional processing and assumptions to reconstruct surfaces between points, which can introduce artifacts or errors.

Three primary approaches have been identified for adapting the 3D Highlighter model to work with point cloud data:

1. **Differentiable Point Cloud Rendering:** Generation of images where each point is represented as a ball in 3D space using PyTorch3D

2. **Mesh Approximation:** Surface reconstruction from point cloud data using Open3D techniques

3. **Voxel-based Conversion:** Transformation of point clouds to voxels and subsequently to voxel meshes using Kaolin ops conversions

3.2.1 Analysis of Approaches

After thorough evaluation, the differentiable point cloud renderer (approach 1) emerges as the optimal solution. This determination is supported by the following advantages:

- **Geometric Preservation:** Maintains the original point cloud structure while enabling direct visualization and optimization of the highlighting process.
- **Direct Processing:** Avoids potential information loss or artifacts that might be introduced through mesh reconstruction or voxelization.

3.2.2 Limitations of Alternative Approaches

Surface reconstruction methods (approach 2) have notable drawbacks, including the potential introduction of geometric errors or artifacts, a strong dependence on point cloud density and distribution for quality and additional computational overhead without significant benefits. Similarly, the voxelization approach (approach 3) is constrained by the discretization of continuous 3D space, which leads to a loss of fine geometric details. It also demands higher memory usage and computational resources, with quality trade-offs that are heavily resolution-dependent.

3.2.3 Alignment with Original Methodology

The differentiable point cloud renderer approach aligns with the original 3D Highlighter methodology through direct CLIP compatibility via 2D image rendering and preservation of original point cloud geometry. This strategy necessitates minimal architectural modifications while maintaining core highlighting capabilities. The approach establishes a robust foundation for extending the 3D Highlighter to point cloud data while preserving essential functionality and performance characteristics.

4. Experiments

In this section, we detail the methodology and outcomes of three key experiments designed to evaluate our pipeline's ability to generate affordance labels without direct supervision, leveraging the 3D AffordanceNet dataset.

4.1. Dataset and Experimental Setup

We utilized the 3D AffordanceNet dataset, which includes 22,949 shapes across 23 semantic classes and 18 affordance types. Our focus was on specific household classes and affordances related to hand-object interactions. This setup is common in the three experiments we will see in this section.

Setup:

- **Subsets Definition:** We defined two subsets from the training set: a validation set for hyperparameter tuning and a test set for evaluating generalization.
- **Prompt Strategies:** Strategies ranged from basic descriptions to affordance-specific phrases to guide affordance highlighting.
- **Grid Search and Hyperparameter Exploration:** A grid search approach was employed to systematically explore various hyperparameters, including learning rates, network depths, number of views, thresholds, number of augmentations, and prompt strategies, with models trained for a fixed number of iterations 500 to evaluate their impact on model performance and optimize affordance segmentation.
- **Evaluation Pipeline:** Using the defined subsets, we evaluated the model first on the validation set to optimize hyperparameters using grid search. To determine the best-performing configurations, we used both single-threshold IoU and the arithmetic mean IoU (aIoU), calculated across thresholds ranging from 0 to 0.99 in 0.01 increments. The aIoU provided a robust evaluation metric by averaging performance across multiple thresholds. The test set was then used to measure generalization with a fixed threshold, with visual inspections and quantitative metrics guiding the final analysis.
- **Visual Assessment:** Ground truth renders were compared with model outputs to handpick the best-performing configurations for further evaluation.

4.2. Experiment 1: Single Object and Affordance Pair

Objective: Evaluate the model's performance on a single object (knife) with a single affordance (cut).

Methodology: A grid search of hyperparameters and prompt strategies was conducted, with each configuration trained for a fixed number of 500 iterations. Three prompt strategies were employed:

- **Basic:** *"A 3D render of a gray Knife with highlighted cut regions."*
- **Action:** *"A 3D render indicating the parts of the gray knife that can be used to cut."*
- **Affordance-Specific (Cut):** *"A 3D render of a gray knife with highlighted regions showing the sharp blade edge and cutting tip, emphasizing the main cutting surface and pointed end."*

Config	Shape	Prompt Strategy	Threshold	Learning Rate	Depth	num. aug	Views	IoU	aIoU
Config 1	d7	Basic	0.01	0.001	4	1	2	0.938	0.109
Config 2	24	Affordance-Specific	0.94	0.001	4	1	2	0.394	0.391
Config 3	**1e**	**Action**	**0.1**	**0.001**	**4**	**3**	**4**	**0.703**	**0.117**
Config 4	3a	Action	0.02	0.001	4	3	2	0.712	0.044
Config 5	**d7**	**Basic**	**0.1**	**0.001**	**4**	**3**	**2**	**0.972**	**0.1389**

Table 2. Top-performing hyperparameter configurations and performance (IoU and aIoU) on the validation set for the *cut* affordance.

4.2.1 Validation Set Observations

As summarized in Tab. 2, the validation set consisted of 5 objects, each representing the *cut* affordance for the *knife* class. A total of 240 renders were evaluated using IoU at specific thresholds and the arithmetic mean IoU (aIoU) over thresholds ranging from 0 to 0.99 in 0.01 increments. Many configurations achieved their highest IoU at low thresholds (e.g., 0.01), which often resulted in excessive highlighting of high-probability regions. While these values could appear promising, they often failed to capture meaningful affordance regions. To address this, aIoU was used as a more robust measure, balancing performance across thresholds.

Visual assessment played a critical role in identifying the best configurations. High IoU or aIoU values did not always align with the ground truth. For example, Config 2 (*Affordance-Specific*) achieved the highest *aIoU* (0.394) but tended to over-highlight large portions of the object. Conversely, some configurations failed to highlight any regions, resulting in low scores. Config 3 (*Action*) and 5 (*Basic*) provided the best balance, achieving *aIoU* values of 0.117 and 0.1389, respectively, with Config 5 also achieving a high IoU (0.972) at a threshold of 0.1.

A notable trend was the peak IoU observed at thresholds between 0.01 and 0.1 (see Fig. 3). To balance precision and recall, a fixed threshold of 0.1 was selected for the test set evaluation. This approach minimizes the inclusion of low-probability regions while retaining confident affordance predictions, ensuring a more reliable evaluation of generalization.

Fig. 4 and Fig. 5 illustrate the ground truth and highlighted renders for Configs 3 and 5, showcasing their strong alignment with the intended affordance regions. These observations highlight the importance of combining quantitative metrics with qualitative visual assessment to identify reliable configurations.

4.2.2 Test Set Observations

The two best-performing hyperparameter configurations from the validation set (*Config 3* and *Config 5*) were tested on 5 unseen objects from the test set, all belonging to the same class (*knife*) and affordance (*cut*) as the validation set. The test aimed to evaluate the generalization ability of these configurations. Predictions showed a mix of out-

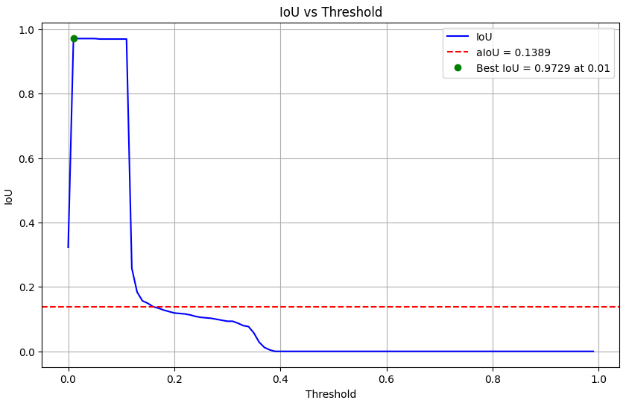


Figure 3. IoU vs. Threshold for Config 5.

(a) Ground Truth Render

(b) Highlighted Render

Figure 4. Comparison of ground truth and highlighted affordance regions for Config 3.

(a) Ground truth render.

(b) Highlighted render.

Figure 5. Comparison of ground truth and highlighted affordance regions for Config 5.

comes, summarized below:

As it can be seen in Fig. 6, for **Config 3** (Action Prompt), 3 out of 5 objects had well-highlighted affordance regions with decent IoU values, indicating good generalization to other knife types. This was a significant improvement compared to the validation phase, where only 1% of renders had decent affordance highlighting. Object 3 achieved the best IoU (IoU = 0.8630) and the mean IoU for Config 3 across the test set was calculated as **0.339**. The results can be seen in Tab. 3.

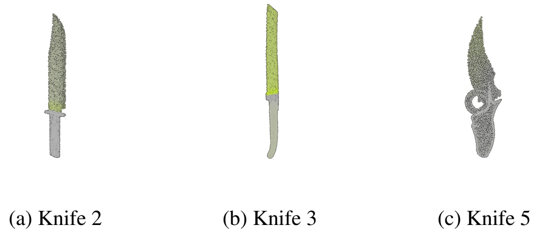

(a) Knife 2 (b) Knife 3 (c) Knife 5

Figure 6. Affordance highlights for three objects predicted using Config 3.

Object	**IoU**
Object 1	0.0000
Object 2	0.4820
Object 3	0.8630
Object 4	0.0000
Object 5	0.3534
Mean IoU	**0.339**

Table 3. IoU results for the five test objects, with a calculated Mean IoU of 0.339.

In contrast, **Config 5** (Basic Prompt) did not perform as well. Only one object exhibited decent highlighting, while the rest were either not highlighted at all or had irrelevant regions (e.g., the knife handle) highlighted. This confirmed that the Action Prompt combined with the selected hyperparameters in Config 3 was more effective for affordance detection in this experiment.

4.3. Experiment 2: Single Object Class with Multiple Affordances

Objective: Evaluate the model's ability to generalize across multiple affordances (*Openable*, *Pushable*, *Pull*) within the *door* class.

Methodology: This experiment required segmenting multiple affordances within the same object class in a shared context.

Two prompt strategies were employed:

- **Basic:** *"A 3D render of a gray door with highlighted {affordance_type} regions."*
- **Affordance-Specific:**
  - *Openable: "A 3D render of a gray door with highlighted hinge regions and handle areas that enable opening movement."*
  - *Pushable: "A 3D render of a gray door with highlighted flat surface regions designed for pushing."*
  - *Pull: "A 3D render of a gray door with highlighted regions showing handles, grip spots, or edges used for pulling."*

4.3.1 Validation Set Observations

We evaluated over 640 renders generated from the grid search with various hyperparameters and prompt strategies. The results were underwhelming. The affordance-specific prompt failed to outperform the basic prompt and often struggled to produce meaningful affordance highlights. Many renders either showed no highlighted regions or over-highlighted the entire door surface, leading to poor IoU and aIoU scores. This likely stemmed from the small, localized nature of these affordances and the model's difficulty capturing precise spatial features like handles or hinges.

Additionally, the general nature of affordance terms such as *pushable* likely contributed to overly broad or inaccurate predictions. For example, the model often interpreted *push* as applicable to the entire door surface, further reducing its ability to localize specific affordance regions. Only a small subset of renders showed average performance. The best configurations for each affordance, summarized in Tab. 4, provided relatively better results but were still suboptimal overall. Due to the limited success in the validation set, all three selected configurations were tested on the test set to

(a) Ground Truth (b) Highlighted

Figure 7. Comparison of ground truth and highlighted affordance regions for Config 2.

Config	Affordance	Prompt Strategy	Threshold	Learning Rate	Depth	Num. Aug	Views	IoU / aIoU
Config 1	Pushable	Basic	0.3	0.0001	5	3	2	0.215 / 0.1623
Config 2	**Openable**	**Basic**	**0.1**	**0.0001**	**5**	**3**	**2**	**0.6694 / 0.3837**
Config 3	Pull	Basic	0.1	0.001	4	3	2	0.6670 / 0.6619

Table 4. Top-performing configurations for the affordances *pushable*, *openable*, and *pull* on the door class.

further evaluate their generalization capabilities. Fig. 7 illustrate the ground truth and highlighted renders for Config 2.

4.3.2 Test Set Observations

The best-performing configurations for each affordance (*pushable*, *openable*, *pull*), identified using the *basic* and *affordance-specific* prompts, were tested on 5 unseen objects in the test set. However, the results were disappointing, confirming the challenges observed during validation.

Most renders were either unhighlighted or misaligned with the intended regions, underscoring the difficulty of generalizing affordances requiring precise spatial localization. The diverse geometries in the test set likely compounded this issue. The mean IoU (mIoU) values were: **Pull** (*0.0000*), **Pushable** (*0.0930*), and **Openable** (*0.0270*). Even the *pushable* affordance, with the highest mIoU, showed largely misaligned predictions upon visual inspection.

These results highlight the limitations of CLIP-based affordance detection for localized affordances. Despite optimized hyperparameters and prompt strategies, the combination of diverse geometries, small affordance regions, and general affordance terms posed significant challenges to accurate predictions.

4.4. Experiment 3: Single Affordance Across Multiple Classes

Objective: Evaluate the model's ability to generalize a single affordance (*contain*) across diverse object classes (*vase*, *bowl*, *bottle*).

Methodology: We conducted a grid search to optimize hyperparameters, utilizing the following three prompt strategies:

- **Basic:** *"A 3D render of a gray {shape_class} with highlighted contain regions."*
- **Action:** *"A 3D render indicating the parts of the gray {shape_class} that can be used to contain."*
- **Affordance-Specific:** *"A 3D render of a gray {shape_class} highlighting the internal cavity or surface regions designed for containing objects."*

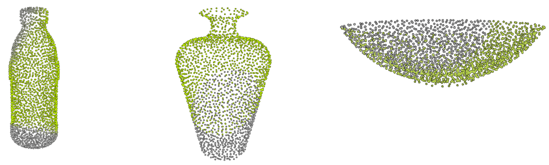
4.4.1 Validation Set Observations

We evaluated over 412 renders using IoU and aIou along with visual assessment. The top-performing configurations, summarized in Tab. 5, revealed better highlighting performance compared to previous experiments. However, this improvement can be attributed to the *contain* affordance, which requires highlighting an object's internal cavity or entire surface, making it less complex than affordances needing localized precision. As a result, predictions for this affordance often aligned well with the ground truth, leading to higher IoU scores.

The affordance-specific prompt consistently outperformed basic and action prompts across all shape classes, providing the highest IoU and aIoU values. Its detailed descriptions of internal cavities or containing surfaces enabled more accurate predictions aligned with the ground truth.

Among the shape classes, the bottle configuration achieved the best results (*IoU* = 0.9043, *aIoU* = 0.6939), with precise and semantically aligned highlights confirmed through visual inspection. Configurations for the vase also performed well but were less consistent.

These findings demonstrate the effectiveness of the affordance-specific prompt for this experiment. The geometry and characteristics of each shape class also influenced performance. Based on these results, the bottle configuration was selected for testing generalization on the test set. Fig. 8 illustrates examples of the ground truth and highlighted renders for the best-performing configurations across the three shape classes.



(a) Bottle (b) Vase (c) Bowl

Figure 8. Comparison of highlighted affordance regions for the *contain* affordance across bottle, vase, and bowl classes.

Shape Class	Prompt Strategy	Threshold	Learning Rate	Depth	Num. Aug	Views	IoU / aIoU
Bottle	**Affordance-Specific**	**0.1**	**0.001**	**4**	**3**	**2**	**0.9043 / 0.6939**
Vase	Affordance-Specific	0.1	0.001	4	1	3	0.652 / 0.648
Bowl	Affordance-Specific	0.1	0.001	5	1	3	0.9155 / 0.295

Table 5. Top-performing configurations for the affordance *contain* across bottle, vase, and bowl classes.

4.4.2 Test Set Observations

The best-performing configuration was tested on 5 test objects across three classes to evaluate generalization. Results revealed significant limitations in generalizing the *contain* affordance across classes.

Most renders were unhighlighted or incorrectly highlighted. Only the bottle class, which formed the basis for the best validation configuration, exhibited any affordance detection. However, its predictions were subpar, with a mean IoU of 0.345 at a threshold of 0.13. The other classes (*vase*, *bowl*) showed no meaningful affordance detection, with near-zero IoU values.

These findings suggest that while the affordance-specific prompt and hyperparameters were effective for bottles, they failed to generalize to other classes like bowl and vase. The model's reliance on bottle-specific geometries and the variation in shapes across classes posed significant challenges.

In summary, the model demonstrated minimal success for the bottle class but failed to generalize the *contain* affordance to other shape classes. Fig. 9 shows the ground truth and predicted highlights for the bottle class, the only one with detectable affordance regions.

(a) Ground Truth

(b) Predicted

Figure 9. Ground truth vs. predicted highlights for the bottle class in the test set; other classes showed no meaningful results.

4.5. Results and Discussion

The experiments revealed key challenges in affordance detection. In Experiment 1, the *action* prompt with optimized hyperparameters showed good generalization for the *cut* affordance in the *knife* class, achieving a mean IoU of 0.542 on the test set. Experiment 2 highlighted poor generalization for multiple affordances (*pushable*, *openable*, *pull*) in the *door* class, with very low mean IoU values, emphasizing the difficulty of localized precision. Experiment 3 demonstrated limited generalization of the *contain* affordance across shape classes (*bottle*, *vase*, *bowl*), with only the *bottle* class yielding meaningful results, albeit subpar. Overall, localized affordances struggled with generalization, particularly across diverse geometries, and while affordance-specific prompts occasionally improved performance, they were inconsistent. The model's reliance on training geometries further limited its ability to generalize across different object classes.

5. Extension of the pipeline

We explore several approaches to enhance model performance through methodological variations. Our investigation includes: evaluating dynamic background integration during rendering, implementing multiple augmentation techniques for improved robustness, testing alternative backbone architectures, and validating with real-world LiDAR point cloud data. This systematic evaluation assesses both model generalization across different scenarios and practical viability in real-world applications.

5.1. Adding background and augmentation

We investigate two optimization strategies: background diversity and image augmentation. For backgrounds, we tested four types (two outdoor, two indoor) replacing the standard black background. For augmentation, we implemented four techniques:

- **Balanced Transform:** Moderate augmentations combining random cropping (95-100%), perspective distortion (p=0.7, scale=0.3), and color adjustments (±10%).
- **Viewpoint Transform:** Geometric transformations with rotations (±15°), translations (±10%), scale variations (±10%), and perspective distortions (p=0.9, scale=0.6).
- **Lighting Transform:** Color manipulations (±20%), grayscale conversion (p=0.05), and Gaussian blur (σ=0.1-0.5).
- **Default Transform:** Minimal augmentation with fixed-scale cropping and perspective distortion (p=0.8, scale=0.5).

Augmentation Type	Background	mIoU
Balanced	No background	0.4924
	outdoor 1	0.3011
	outdoor 2	0.3430
	indoor 1	0.1657
	indoor 2	0.1861
Viewpoint	No background	0.5360
	outdoor 1	0.3330
	outdoor 2	0.4790
	indoor 1	0.6639
	indoor 2	0.3654
Lighting	No background	0.5881
	outdoor 1	0.3342
	outdoor 2	0.5799
	indoor 1	0.4261
	indoor 2	0.0669
Default	No background	0.6790
	outdoor 1	0.3263
	outdoor 2	0.4708
	indoor 1	0.3851
	indoor 2	0.1921

Table 6. Comparison of augmentation strategies and background types by mIoU, highlighting their varying effectiveness.

We applied these techniques to the *Knife* class with *Cut* affordance using *Config 5*, measuring *mean IOU* over 3 shapes.

Results in Tab. 6, show that *default* augmentation achieves highest mIoU (0.6790) with no background, while *viewpoint* augmentation performs best with indoor backgrounds (0.6639). Generally, adding backgrounds decreases performance, with outdoor backgrounds performing slightly better than indoor ones. These findings suggest simpler backgrounds yield better results for affordance detection systems.

5.2. Alternative Backbones

We explored the potential benefits of using OpenCLIP with the ViT-H-14-378-quickgelu architecture [2], which was trained on the extensive DFN-5B dataset, as an alternative backbone to the original CLIP model. Despite the larger-scale training data and more sophisticated architecture, this variant did not yield performance improvements over the baseline CLIP model. The results showed consistently lower performance metrics; thus, we omit the detailed numerical results as they provide no additional insights into model enhancement. Additionally, we investigated the possibility of incorporating OpenShape [5], a model specifically designed for direct 3D point cloud processing, which could potentially eliminate the need for intermediate rendering steps in our pipeline. However, due to computational resource constraints and hardware limitations, we were unable to successfully integrate this model into our framework. This remains an interesting direction for future work, particularly in scenarios where more substantial computational resources are available. These experiments, while not yielding immediate improvements, provide valuable insights into the robustness of the original CLIP architecture for our specific task and highlight important considerations for future research directions in 3D affordance detection.

5.3. LiDAR

To validate our pipeline with real-world data, we collected point cloud data using an iPhone 16 Pro's LiDAR sensor through the KIRI Engine application (see Fig. 10). We tested the model's ability to highlight the *sit* affordance on a standard office chair using the action prompt: *"A 3D render indicating the parts of the gray chair that can be used to sit."* The results demonstrate our pipeline's adaptability to real-world LiDAR-acquired point clouds.

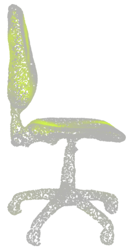


Figure 10. 3D scan of chair using iPhone 16 Pro LiDAR.

6. Conclusion

This work explores adapting the 3D Highlighter architecture for label-free affordance detection in point clouds, highlighting key challenges and opportunities. Our systematic experimentation across different object classes and affordance types revealed that performance is heavily influenced by hyperparameter configuration and prompt engineering strategies. Through extensive ablation studies, we demonstrated that while the approach shows promise for certain scenarios, significant challenges remain in achieving robust, generalizable affordance detection.

Future work should focus on improving cross-class generalization, developing more sophisticated prompt engineering strategies, and investigating alternative neural architectures specifically designed for 3D affordance understanding. Additionally, exploring ways to incorporate geometric priors while maintaining the label-free nature of the approach could potentially enhance performance on complex, localized affordances.

References

[1] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. 1

[2] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 2818–2829. IEEE, June 2023. 8

[3] Dale Decatur, Itai Lang, and Rana Hanocka. 3d highlighter: Localizing regions on 3d shapes via text descriptions. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20930–20939, 2023. 1

[4] Sheng Deng, Xun Xu, Chaozheng Wu, Ke Chen, and Kui Jia. 3d affordancenet: A benchmark for visual object affordance understanding. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1778–1787, 2021. 1

[5] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding, 2023. 8

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 1

[7] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 41(1):1–12, 2022. 1

[8] Yuhang Yang, Wei Zhai, Hongchen Luo, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Grounding 3d object affordance from 2d interactions in images, 2023. 2