# Grounding 3D Object Affordance with Language Instructions, Visual Observations and Interactions

Supplementary Material

| Tensor | Dimension | Meaning |
|---|---|---|
| $I_I$ | $3 \times 224 \times 224$ | Image |
| $I_P$ | $3 \times 2048$ | Point cloud |
| $F_{2D}$ | $512 \times 7 \times 7$ | 2D feature |
| $F_{3D}$ | $512 \times 64$ | 3D feature |
| $F_S$ | $113 \times 512$ | Spatial feature |
| $F_{SP}$ | $113 \times 4096$ | Project $F_S$ to a semantic space |
| $F_T$ | $N_L \times 4096$ | Textual feature |
| $F_A$ | $113 \times 512$ | Affordance feature |
| $O$ | $2048 \times 1$ | 3D object affordance |

Table 1. **Description of tensors.** We present the dimensions and meanings of the input and output tensors in each component of the model. Here we do not specify the batch size.

## A. Implementation Details

**Dataset Details.** In our dataset, the image resolutions vary, while the point clouds have been post-processed to ensure each object contains exactly 2048 points. In the seen setting, both the training and testing sets contain 23 object categories. In the unseen setting, there are 17 object categories in the training set and 6 categories in the testing set.

- In the **seen** setting, the object categories in both the **training** and **testing** sets are: bag, bed, bottle, bowl, chair, clock, dishwasher, display, door, earphone, faucet, hat, keyboard, knife, laptop, microwave, mug, refrigerator, scissors, storage furniture, table, trashcan, and vase.
- In the **unseen** setting, the object categories in the **training** set are: bag, bottle, bowl, chair, clock, display, door, earphone, faucet, hat, keyboard, knife, mug, refrigerator, storage furniture, table, and trashcan. The object categories in the **testing** set are: bed, dishwasher, laptop, microwave, scissors, and vase.

**Method Details.** For data augmentation during training, we randomly crop and resize the input images to $3 \times 224 \times 224$. Then both the images and point clouds are normalized. For the 2D vision encoder, the 2D features it outputs are reshaped into $512 \times 49$ and fused with the 3D features. For the 3D vision encoder, we use 3 set abstraction layers with multi-scale grouping to extract point-wise features, respectively sampling 512, 128, and 64 points per layer. The embedding dimension is set to 512. For the vision-language model, the base model is vicuna-7b[8] and its hidden size is set to 4096. The token length ($N_L$) varies with the instructions, and its maximum length is capped at 32. The weight of focal loss ($\omega_f$) and dice loss ($\omega_d$) are both set to 1. The dimensions of tensors in the whole pipeline are shown in the Tab. 1.


Figure 1. **Different shapes.** The objects in the instructions, images and point clouds belong to the same category but have different geometries.


Figure 2. **Different categories.** The objects in the instructions, images and point clouds have different categories and geometries. **(Row 1:)** The category of object in the instruction and image is "refrigerator", while the object of point cloud is "storage furniture". **(Row 2:)** The category of object in the instruction and image is "bag", while the object of point cloud is "hat".

## B. Additional Experiments

### B.1. Results

**Different Backbones.** To evaluate the impact of different backbones on model performance, we select various backbone networks. For 2D vision encoder, since the standard LLaVA[2–4] uses the CLIP image encoder, we compare CLIP-ViT[6] with the ResNet18[1] we utilize. For 3D vision encoder, we additionally select PointConvFormer[7] for comparison. We calculate the FLOPs and parameter counts of different backbones: ResNet18 (1.82 GFLOPs, 11.18 M params), PointNet++ (4.94 GFLOPs, 0.51 M params), CLIP-ViT (51.90 GFLOPs, 202.05 M params), PointConvFormer (17.06 GFLOP, 2.32 M params). As shown in the Tab. 2, although larger backbone networks generally lead to better model performance, the improvement is not significant. Therefore, to make the model more efficient and lightweight, we choose ResNet18 and PointNet++[5] as the final backbones.

| Setting | Metrics | Full-view | | | Partial-view | | | Rotation-view | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Baseline | ViT | PCF | Baseline | ViT | PCF | Baseline | ViT | PCF |
| Seen | AUC ↑ | 0.8895 | 0.8986 | 0.8929 | 0.8478 | 0.8516 | 0.8497 | 0.7823 | 0.7832 | 0.7829 |
| | IOU ↑ | 0.2123 | 0.2211 | 0.2102 | 0.1755 | 0.1782 | 0.1768 | 0.1161 | 0.1170 | 0.1165 |
| | SIM ↑ | 0.6102 | 0.6183 | 0.6114 | 0.5928 | 0.5984 | 0.5950 | 0.5191 | 0.5206 | 0.5207 |
| | MAE ↓ | 0.0816 | 0.0784 | 0.0815 | 0.0921 | 0.0898 | 0.0915 | 0.1182 | 0.1159 | 0.1172 |
| Unseen | AUC ↑ | 0.7741 | 0.7849 | 0.7767 | 0.7602 | 0.7676 | 0.7611 | 0.6303 | 0.6340 | 0.6311 |
| | IOU ↑ | 0.0903 | 0.0984 | 0.0893 | 0.0724 | 0.0769 | 0.0725 | 0.0415 | 0.0477 | 0.0438 |
| | SIM ↑ | 0.4089 | 0.4170 | 0.4099 | 0.4144 | 0.4198 | 0.4140 | 0.3842 | 0.3869 | 0.3841 |
| | MAE ↓ | 0.0945 | 0.0878 | 0.0939 | 0.1183 | 0.1042 | 0.1174 | 0.1398 | 0.1358 | 0.1395 |

Table 2. **Different backbones.** Here we present the results of models using different backbones under various views and settings. Among them, "Baseline" refers to using ResNet18 and PointNet++ as the backbone, "ViT" refers to using CLIP-ViT and PointNet++ as the backbone, and "PCF" refers to using ResNet18 and PointConvFormer as the backbone.

| Setting | Metrics | Full-view | | | Partial-view | | | Rotation-view | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| Seen | AUC ↑ | 0.8786 | 0.8895 | 0.8880 | 0.8435 | 0.8478 | 0.8462 | 0.7531 | 0.7823 | 0.7842 |
| | IOU ↑ | 0.2055 | 0.2123 | 0.2098 | 0.1733 | 0.1755 | 0.1803 | 0.1008 | 0.1161 | 0.1188 |
| | SIM ↑ | 0.5958 | 0.6102 | 0.6115 | 0.5908 | 0.5928 | 0.5919 | 0.5006 | 0.5191 | 0.5196 |
| | MAE ↓ | 0.0861 | 0.0816 | 0.0824 | 0.0923 | 0.0921 | 0.0940 | 0.1262 | 0.1182 | 0.1204 |
| Unseen | AUC ↑ | 0.7598 | 0.7741 | 0.7858 | 0.7586 | 0.7602 | 0.7631 | 0.5955 | 0.6303 | 0.6394 |
| | IOU ↑ | 0.0879 | 0.0903 | 0.0910 | 0.0698 | 0.0724 | 0.0723 | 0.0394 | 0.0415 | 0.0457 |
| | SIM ↑ | 0.3911 | 0.4089 | 0.4004 | 0.4105 | 0.4144 | 0.4135 | 0.3807 | 0.3842 | 0.3860 |
| | MAE ↓ | 0.1012 | 0.0945 | 0.0955 | 0.1334 | 0.1183 | 0.1166 | 0.1436 | 0.1398 | 0.1421 |

Table 3. **Different pairings.** We show the results when the number of pairings varies under different views and settings in detail. One image could be paired with multiple point clouds during training. The number of pairings has an influence on the model performance.

**Different Pairings.** Since the images and point clouds in the dataset come from different physical instances, matching a single image with multiple point clouds can increase the diversity of the data. Here, the number of pairings is set to 1, 2, and 3, with the results shown in the Tab. 3. When the number of pairings is 3, the batch size is set to 4 due to memory limitations. And the training time has increased. From the results in the table, it can be seen that the model performance improves significantly when the number of pairings is changed from 1 to 2, while the performance improvement from 2 to 3 pairings is much smaller and even some metrics even decrease. Considering the above results, we finally set the number of pairings to 2 in our implementation.

## B.2. Visualization

**Mismatch.** What will happen when the objects in the instruction, image and point cloud are different? To explore this issue, we perform experiments with different shapes or categorios. As shown in the Fig. 1 and Fig. 2, they indicate that the model has mapped the cross-category invariance between affordance and geometric shapes, enabling generalization to new geometric instances. However, when the object categories differ, the model tends to follow the instructions and does not predict affordances for different objects.

**Multiplicity.** A single object can have multiple affordances. To evaluate multiplicity, Fig. 4 shows the different results that the model predicts based on the same image and point cloud when given different language instructions. From the results, it can be seen that the model performs well in instruction-following, which allows it to interact with users and adapt to different scenarios.


Figure 3. **Failure Cases.** In the rotation-view and unseen setting, **(Row 1)** the model fails to ground the dishwasher door handle and only ground the door itself; **(Row 2)** the model is not very effective at grounding in small affordance regions, such as the buttons on the microwave.

**Limitations.** The model has limitations with rotated views in unseen scenarios, as shown in the Fig. 3. This may be because the model needs to be designed to extract features which are invariant to rotation and symmetry for this situation. Additionally, the model has a limited understanding of fine-grained geometric parts and may be necessary to further increase the receptive field.

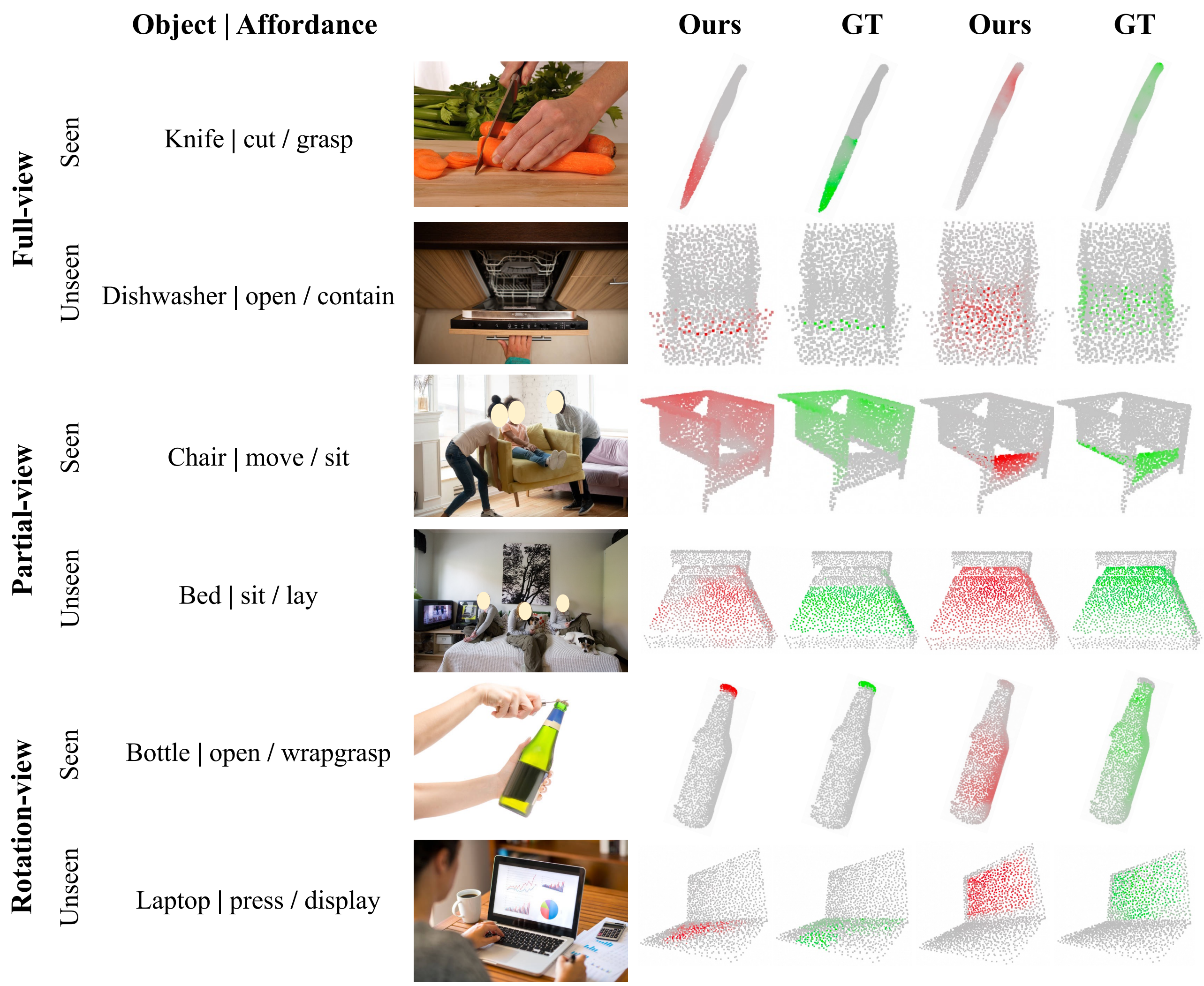


Figure 4. **Affordance multiplicity.** We keep the image and point cloud inputs unchanged while modifying the language instructions to demonstrate the model's instruction-following capability. In the figure, we have omitted the instructions, retaining only the object and different affordance names.

## References


[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[2] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023. 1

[3] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[4] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024. 1

[5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1

[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. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[7] Wenxuan Wu, Li Fuxin, and Qi Shan. Pointconvformer: Revenge of the point-based convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21802–21813, 2023. 1

[8] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023. 1