CatFree3D: Category-Agnostic 3D Object Detection With Diffusion (Supplementary Material)

Wenjing Bian[1] Zirui Wang[1] Andrea Vedaldi[2]
[1]Active Vision Lab [2]Visual Geometry Group
University of Oxford
{wenjing, ryan, vedaldi}@robots.ox.ac.uk
https://bianwenjing.github.io/CatFree3D

A. Implementation Details

A.1. Prompt Encoding

We encode the image I, a 2D bounding box B, camera intrinsics K, and the object depth z into the conditioning signal $\mathbf{c}$ through

$$\mathbf{c} = g(I, B, K, z), \tag{1}$$

The 2D bounding box B is described by the centre of the box along with its height and width on the image plane, i.e.

$$B := [u_{\text{2d}}, v_{\text{2d}}, w_{\text{2d}}, h_{\text{2d}}] \tag{2}$$

To account for variation in depth and focal length, we further unproject the width and height of the 2D box into 3D using the following equation:

$$(w_{\text{3d}}, h_{\text{3d}}) = (w_{\text{2d}}\frac{z}{f_x}, h_{\text{2d}}\frac{z}{f_y}), \tag{3}$$

where f_x and f_y are the focal lengths from the intrinsics K.

For the input image I, we first encode it with a pre-trained Swin Transformer [3] to generate multi-scale feature maps F. Next, we extract local image features inside the region of the 2D box prompt to obtain F_{RoI}. Additionally, we apply a cross-attention layer [6] between F and the 2D box B to obtain F_{atten}.

By concatenating the transformed box prompt, image features and the object depth, the final conditioning signal $\mathbf{c}$ can be written as

$$\mathbf{c} = [F_{\text{RoI}}, F_{\text{attn}}, u_{\text{2d}}, v_{\text{2d}}, w_{\text{3d}}, h_{\text{3d}}, z]. \tag{4}$$

A.2. Loss Function

The Chamfer distance between the corners of the predicted 3D boxes $\mathcal{M}_{\text{pred}} = \{a_i | i = 1...8\}$ and the corners of the ground truth boxes $\mathcal{M}_{\text{gt}} = \{b_i | i = 1...8\}$ is computed as

$$\mathcal{L}_{\text{chamfer}} = \sum_{a_i \in \mathcal{M}_{\text{pred}}} \min_{b_i \in \mathcal{M}_{\text{gt}}} \|a_i - b_i\|_1 + \sum_{b_i \in \mathcal{M}_{\text{gt}}} \min_{a_i \in \mathcal{M}_{\text{pred}}} \|b_i - a_i\|_1. \tag{5}$$

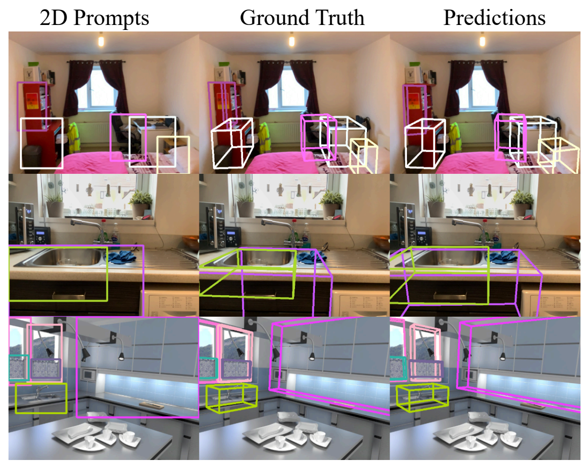

Figure 1. **Generalisation Performance: Results for Cross-Dataset Test.** We show predictions on ARKitScenes and Hypersim made by our method trained on SUN RGB-D.

A.3. Baseline Models

Unprojection Fig. 3 illustrates how we obtain the Unprojection baseline for experiments in Sec.5.3 of the main paper.

Total3DUnderstanding [4] We use their publicly released code and the model pre-trained on SUN RGB-D in experiments of Section 5.

Cube R-CNN [2] For the results on the SUN RGB-D dataset in the second row of Table 1 in the main paper, we use the numbers reported directly from their paper. For the other experiments in Section 5, we use their publicly available code and pre-trained models.

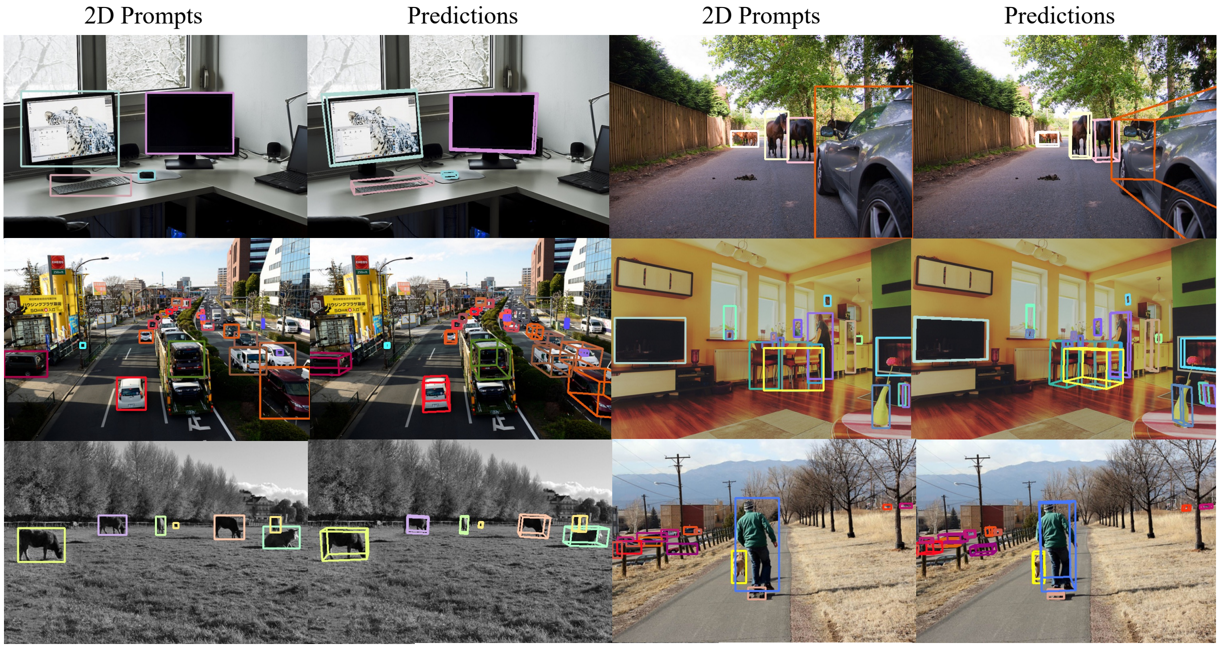

Figure 2. **Generalisation Performance: Additional Results for In-the-Wild Objects on COCO Dataset.** We show predictions made by our method without knowing object depths or camera intrinsics. By using constant values for depths and camera intrinsics, our approach accurately predicts 3D boxes with well-aligned projections on the image.

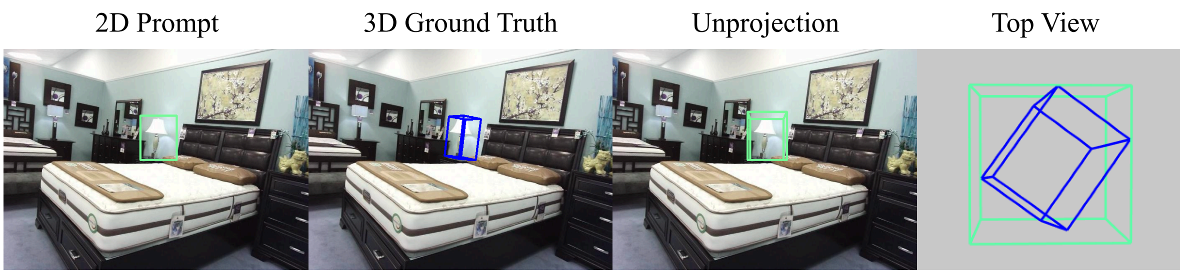

Figure 3. **Unprojection Baseline Illustration.** The Unprojection baseline (green) converts GT 2D boxes to 3D using GT depth and dimensions that match the GT 3D box (blue), with the 3D rotation to zero degrees.

A.4. Algorithms

The training and inference algorithms are shown in Algorithm 1 and Algorithm 2.

B. Additional Results

B.1. Generalisation

Cross-dataset generalisation Our model trained on SUN RGB-D achieves an average IoU of 39.0 on the Hypersim [5] test set and 48.2 on the ARKitScenes [1] test set, highlighting its strong generalisation across different datasets. Fig. 1 presents some of the test results.

Additional COCO results Fig. 2 shows additional results on COCO dataset.

Table 1. **Randomness Analysis on SUN RGB-D test set.** We evaluate the model using 10 different random seeds and report the mean, maximum, minimum, and standard deviation σ for both IoU and NHD.

	Mean	Max	Min	σ
IoU (%) $\uparrow$	61.38	61.46	61.24	6.4e-4
NHD $\downarrow$	0.1140	0.1146	0.1133	3.9e-4

B.2. Per-category SUN RGB-D Performance

In Table 1 of the main paper, we report the average IoU and NHD on 10 common categories of the SUN RGB-D dataset to make a fair comparison for baseline methods with different categories. Tab. 2 and Tab. 3 show the per-category IoU and NHD performances respectively.

Table 2. **Per-category IoU (%) on SUN RGB-D test set.** The top three rows use GT 2D boxes along with predicted depths. The depths of our predictions are set to the same as [2] for fair comparison. The bottom three rows use GT 2D boxes and GT depths for all methods.

Methods	Trained on	table	bed	sofa	bathtub	sink	shelves	cabinet	fridge	chair	toilet	avg.
Total3D	SUN RGB-D	28.0	37.0	30.1	27.6	20.1	10.8	14.3	20.2	24.8	35.4	24.8
Cube R-CNN	SUN RGB-D	39.2	49.5	46.0	32.2	31.9	16.2	26.5	34.7	39.9	45.7	36.2
Cube R-CNN	Omni3D Indoor	41.4	50.9	**50.8**	**39.2**	35.0	17.8	28.2	**35.1**	41.3	48.1	38.8
Ours-d	SUN RGB-D	**42.2**	**54.4**	50.5	38.9	**40.3**	**19.7**	**29.4**	33.5	**43.2**	**50.1**	**40.2**
Total3D*	SUN RGB-D	45.0	47.9	49.7	49.5	44.8	30.8	38.2	48.2	56.3	55.8	46.6
Cube R-CNN*	Omni3D Indoor	54.8	57.0	62.9	52.7	49.7	37.5	47.6	58.5	63.6	61.7	54.5
Ours*	SUN RGB-D	**63.1**	**64.3**	**64.8**	**56.7**	**62.6**	**44.0**	**56.5**	**62.2**	**70.3**	**68.9**	**61.4**

Table 3. **Per-category NHD on SUN RGB-D test set.** The top three rows use GT 2D boxes along with predicted depths. The depths of our predictions are set to the same as [2] for fair comparison. The bottom three rows use GT 2D boxes and GT depths for all methods.

Methods	Trained on	table	bed	sofa	bathtub	sink	shelves	cabinet	fridge	chair	toilet	avg.
Total3D	SUN RGB-D	0.352	0.254	0.314	0.288	0.526	0.497	0.443	0.380	0.408	0.297	0.376
Cube R-CNN	Omni3D Indoor	0.230	0.162	**0.164**	**0.215**	0.244	0.324	0.384	**0.229**	0.233	0.172	0.236
Ours-d	SUN RGB-D	**0.219**	**0.156**	0.167	0.219	**0.231**	**0.322**	**0.372**	0.233	**0.230**	**0.162**	**0.231**
Total3D*	SUN RGB-D	0.204	0.168	0.180	0.157	0.188	0.251	0.210	0.177	0.148	0.160	0.184
Cube R-CNN*	Omni3D Indoor	0.148	0.125	0.107	0.149	0.146	0.181	0.176	0.117	0.107	0.112	0.137
Ours*	SUN RGB-D	**0.114**	**0.107**	**0.101**	**0.120**	**0.114**	**0.161**	**0.127**	**0.111**	**0.093**	**0.090**	**0.114**

B.3. Randomness Analysis

As discussed in Section 5.1 of the main paper, the diffusion process involves inherent randomness, so we conducted the experiments using 10 different random seeds and report the averaged results. To assess the model's stability, in addition to the averaged value reported in the main paper, we also provide the maximum, minimum, and standard deviation across these 10 runs in Tab. 1.

B.4. Noise on 2D Box

We analyse the model's robustness towards noise during inference in Tab. 4. We simulate box noise by applying Gaussian noise to box scales and translations separately, which can be written as:

$$w' = w + \mathcal{N}(0, \sigma^2_{\text{scale}}) \;\; h' = h + \mathcal{N}(0, \sigma^2_{\text{scale}}), \tag{6}$$

$$x' = x + \mathcal{N}(0, \sigma^2_{\text{trans}} \cdot w) \;\; y' = y + \mathcal{N}(0, \sigma^2_{\text{trans}} \cdot h), \tag{7}$$

where w, h are the height and width of the ground truth boxes, x, y are the centre coordinates and w', h', x', y' are the noisy parameters. σ^2_{scale} and σ^2_{trans} are the variances of scale and translation noise. Tab. 4 shows that while the model is robust to noise in box scale and translation, translation errors have a greater impact on accuracy.

Table 4. **Noise on the 2D box**. We add different levels of random noise to the scale and translation of the 2D object box and report the model performance with these noisy box inputs.

σ_{scale}	σ_{trans}	IoU (%) ↑	NHD ↓
0.00	0.00	61.4	0.114
0.05	0.00	59.9	0.120
0.00	0.05	56.1	0.132
0.05	0.05	55.2	0.135
0.10	0.10	46.1	0.174

Algorithm 1: Training

```
def train_loss(images, gt_cubes, boxes_2d):
    """
    images: [B, H, W, 3]
    gt_cubes: [B, 1, D]
    boxes_2d: [B, 4]

    D: dimension of cubes
    N_train: number of sampled boxes during
        training
    """

    # Encode image features
    feats = image_encoder(images)

    # Separate depth information
    # from cube parameters
    cube_params, depths = separate_depth(gt_cubes
        )

    # normalise cube_params to [0, 1]
    cube_params = normalise_cube(cube_params)

    # Duplicate cube_params to N_train
    x_0 = duplicate_cubes(cube_params)

    # Signal scaling
    x_0 = (x_0 * 2 - 1) * scale

    # Corrupt x_0
    t = randint(0, T) # time step
    eps = normal(mean=0, std=1) # noise: [B,
        N_train, D-1]
    x_t = (
        sqrt(alpha_cumprod(t)) * x_0
        + sqrt(1 - alpha_cumprod(t)) * eps
    )

    # Predict
    x_0_pred = denoising_model(
        x_t, feats, t, boxes_2d, depths
    )

    # Set prediction loss
    loss = L(x_0_pred, gt_cubes)

    return loss
```

Algorithm 2: Inference

```
def infer(images, steps, T, boxes_2d, depths):
    """
    images: [B, H, W, 3]
    steps: number of sampling steps
    T: total number of time steps
    boxes_2d: [B, 4]
    depths: object depths [B, 1]

    N_eval: number of proposal boxes during
        inference
    """

    # Encode image features
    feats = image_encoder(images)

    # Initialise noisy cube parameters (excluding
        depth) [B, N_eval, D-1]
    x_t = normal(mean=0, std=1)

    # Define uniform sampling step sizes
    times = reversed(
        linspace(0, T, steps)
    )

    # Generate pairs of consecutive time steps
    time_pairs = list(
        zip(times[:-1], times[1:])
    )

    # Iterate through time pairs
    for t_now, t_next in time_pairs:
        # Predict cube parameters x_0 from x_t
        x_0_pred = denoising_model(
            x_t, feats, t_now, boxes_2d, depths
        )

        # Estimate x_t at t_next
        x_t = ddim_step(
            x_t, x_0_pred, t_now, t_next
        )

    # Combine predicted cube parameters with
        depth information
    pred_cubes = combine_cubes(x_0_pred, depths)

    return pred_cubes
```

References

[1] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, Tal Dimry, Brandon Joffe, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *NeurIPS*, 2021. 2

[2] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In *CVPR*, 2023. 1, 3

[3] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1

[4] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020. 1

[5] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 2

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 1