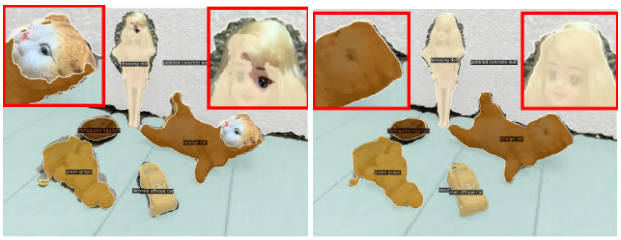
Lift3D: Zero-Shot Lifting of Any 2D Vision Model to 3D

Supplementary Material

Model	ResShift [63]	DDColor [22]	OVSeg [28]
Feature Prediction	3.121	12.282	10.117
Feature Interpolation			
w/o Inconsistency Correction	0.042	7.491	0.375
w/ Inconsistency Correction			
Single Stage	0.0395	3.39	0.371
Ours	0.021	3.240	0.370

Table 3. Ablation study of several components of Lift3D on lifting three unseen 2D feature encoders to 3D. The indent indicates the studied setting is added upon the upper-level ones.



(a) Single Stage (b) Two Stage

Figure 8. Qualitative comparison on the open vocabulary segmentation task between a Single Stage and Two Stage feature correction pipeline.

A. Ablation Studies

We primarily ablate on the following key design considerations of our method.

Feature Prediction. Inspired by [61], we propose a baseline that directly predicts the target view feature and color. To do so, we leverage a powerful generalizable NVS method GNT [50], and add an additional feature head that is supervised by the 2D vision model.

w/o Inconsistency Correction. Next, We remove the inconsistency correction module on the source view features and simply share the aggregation weights between the epipolar RGB features and similarly constructed epipolar projections on the encoder features from the 2D vision model.

Single Stage. Lastly, we convert our two-stage pipeline into a "one-stage" feature correction pipeline using the randomly sampled points along the ray (in contrast to our two-stage, density proposal followed by feature propagation only on the importance-sampled points).

We report performance on the above investigations in Table 3, specifically the mean squared error distance between the estimated feature and ground truth feature obtained by naively encoding the target view using the 2D vision model (or $\boldsymbol{G}_{\text{target}}$). We follow the training strategy discussed in Sec. 5.1 *i.e.* trained on DINO and CLIP features and evaluated on unseen 2D vision models - ResShift [63], DDColor [22], and OVSeg [28].

A straightforward extension of [61], that directly predicts the target view feature cannot generalize to unseen feature encoders resulting in a very high error rate. Instead, our method uses a feature interpolation strategy that derives consistency information from RGB to rectify and propagate inconsistent feature maps. We verify that our inconsistency correction module on the source view features $\{\boldsymbol{G}_i\}_{i=1}^{N}$ is essential and ensures better blending of feature maps. This becomes even more apparent in the case of severely view-inconsistent input features, *e.g.* in the case of the colorization task (DDColor [22]). Finally, our method is two-stage *i.e.* derives a coarse density proposal and performs corrective feature aggregation only on the importance-sampled points. This is necessary and leads to slight deviations in the estimated novel view feature otherwise (and even worse decoded outputs, see Fig. 8).

2D DINO	3D "Lifted" Dino		
N.A.	3 views	6 views	10 views
0.39 / 0.90 / 0.36	0.76 / 0.97 / 0.76	0.80 / 0.98 / 0.80	0.82 / 0.98 / 0.83

Table 4. Effect of the number of input views for semantic segmentation of scenes. Metrics are ordered as IoU / Acc / mAP (higher is better)

B. Computational Efficiency

Naively applying a 2D vision model on each rendered view yields multi-view inconsistent predictions and can be quite inefficient. On the other hand, our method *Lift3D* uses the 2D vision model to encode only the training views (can be pre-computed) and relies on nearest source views when estimating the features from arbitrary viewing angles, that are further decoded to obtain the desired output. This is significantly efficient and faster especially when performing the downstream task from a large number of arbitrary viewpoints.

Formally, let's assume the time to encode, decode, and render each view as t_{enc}, t_{dec}, t_{rend} respectively. To perform the desired task on 100 rendered views, the 2D baseline would roughly take time $t_{\text{2D}} = 100 \times (t_{\text{rend}} + t_{\text{enc}} + t_{\text{dec}})$. Assuming we have around 15 training views for each scene, the time taken by Lift3D $t_{\text{3D}} = 15 \times (t_{\text{enc}}) + 100 \times (t_{\text{rend}} + t_{\text{dec}})$. We can clearly see that $t_{\text{3D}} < t_{\text{2D}}$, and the difference is quite significant in the case of lifting diffusion features like InstructPix2Pix [3] that requires a time-consuming multistep denoising process during encoding *i.e.* $t_{\text{enc}} \gg t_{\text{dec}}$. Therefore, when performing downstream tasks on several arbitrary viewing angles, our method also boasts of superior efficiency along with multi-view consistency when compared to its corresponding 2D counterpart.

C. Limitations

We acknowledge that an interpolation technique like ours from input views does result in some loss in quality, arguably not very significant. However, in several practical

Method	Text-Image Direction Similarity ↑	Direction Consistency ↑
InstructNeRF2NeRF [17]	0.180	0.966
Ours	0.193	0.982

Table 5. Quantitative results for text-guided scene editing.

applications multi-view consistent outputs are usually even more desirable, and even mild deviations from the current viewpoint yield significantly different outputs when naively applying a 2D vision model (see Figures 1, 6, 7, 9). Although our method successfully lifts many 2D features to be multi-view consistent, its potential remains capped by the epipolar-based rendering. For example, our method may not handle sparse 360-degree scenes or objects with complex light transport where epipolar geometry no longer holds and drops in performance with limited number of input views (see Table 4). Interesting future directions include scaling up training of Lift3D to unbounded scenes [8] or combining extant pre-trained 3D models with 2D models.

D. Gallery of Tasks

In Fig. 9, we qualitatively compare the decoded outputs using our 3D lifted features against the original 2D operation on two views and across different tasks. We can clearly see that our method yields multi-view consistent predictions, unlike the 2D operator. In some cases, we even see that our method yields improved predictions perhaps due to multi-view information. For example in Figs. 9k and 9k, we can see that our method is able to segment the hair dryer along with its cable as per the input prompt *"hair dryer with cable"*. Similarly in Figs. 9n and 9o, in the case of super-resolution, we see that our lifted features preserve the original scene geometry with higher detail, unlike its 2D counterpart. For the sake of completeness, we also provide quantitative results for 3D scene editing in Table 5. Following the evaluation protocol in [17], we compute the text-image direction similarity and consistency scores across 6 scenes and report average metrics. Our method outperforms the SOTA scene-specific 3D editing technique InstructNeRF2NeRF [17], both in terms of editing quality and multi-view consistency. Fig. 9 only represents a few tasks and in practice, our method can be extended to any 2D vision operator without any extra tuning.

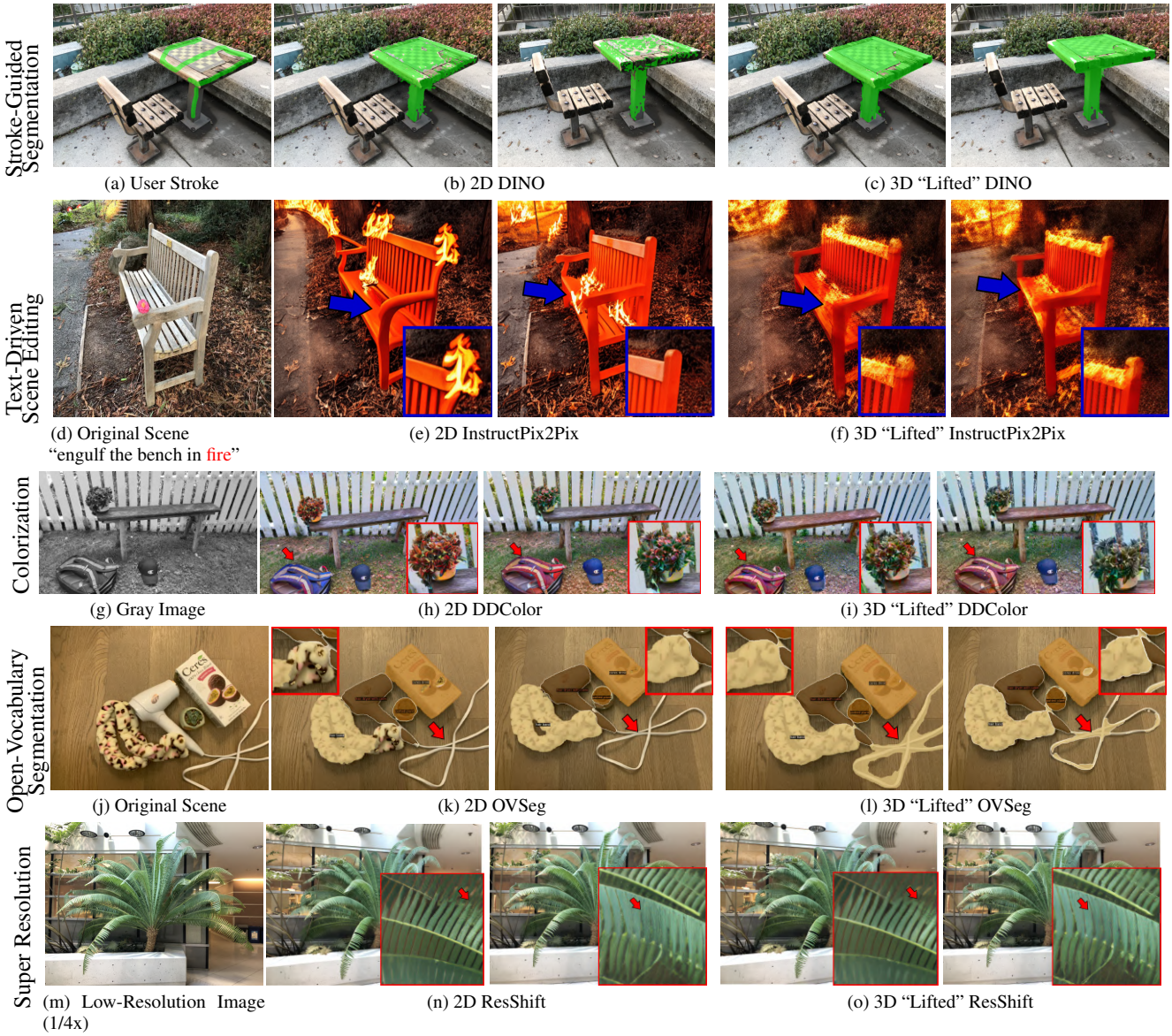


Figure 9. Qualitative comparisons of the 3D "Lifted" features against its corresponding 2D counterpart on two different views and across several tasks. We observe that our 3D-corrected features are more multi-view consistent and sometimes even improve prediction quality. For clearer comparison between the 2D and 3D outcomes, we recommend zooming into the electronic version of this image.