High-Fidelity 3D Textured Shapes Generation by Sparse Encoding and Adversarial Decoding

Supplementary Material

1 Sparse Quantization

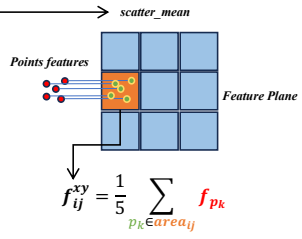
See Supp-Fig. 1.

Supp-Fig. 1. We visualize the input dense point clouds (2880K points, left), low-resolution voxelization result (resolution=100, middle), and high-resolution voxelization result (resolution=1000, right).

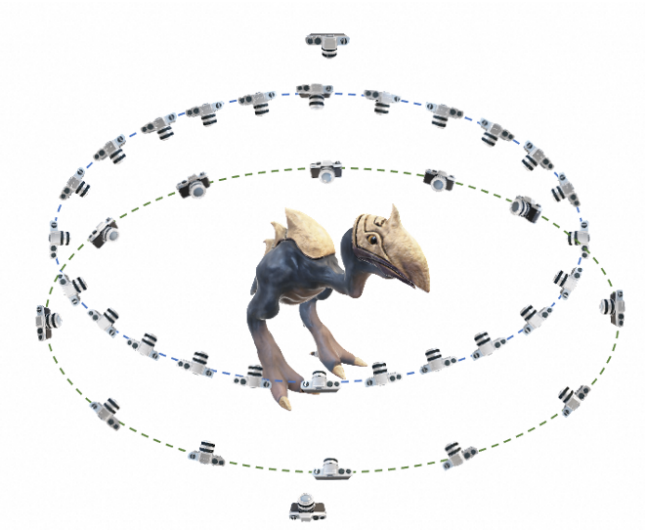
2 Coordinate-based feature scatter

We first extract the feature of each point by SparseConv and then use *TorchScatter* to project the point feature to triplanes with resolution 256. The following code illustrates the process.

```python
from torch_scatter import scatter_mean
def project(self, points_loc, points_feat, plane='xz', triplane_res=256):
    """
    points_loc: B x N x 3 ;  points_feat: B x N x C;
    plane: choose in ['xy','yz','xz'];
    """
    # normalize to the range of (0, 1)
    xy = self.normalize_coordinate(points_loc.clone(), plane=plane, padding=0)
    # acquire indices in triplane space which range in: [0, triplane_res**2).
    index = self.coordinate2index(xy, triplane_res) # B x 1 x N
    # scatter plane features from points
    points_feat = points_feat.permute(0, 2, 1) # B x C x N
    feat_plane = points_feat.new_zeros(points_feat.size(0), C, triplane_res**2)
    # feat_plane: B x C x triplane_res^2
    feat_plane = scatter_mean(points_feat, index, out=fea_plane)
    feat_plane = feat_plane.reshape(p.size(0), C, triplane_res, triplane_res)
    return feat_plane
```



Supp-Fig. 2. ZoomIn for the best view.



Supp-Fig. 3. Rendering configuration of Sparse3D.

3 Dataset Analysis

In Fig. 2, we split Objaverse into ten classes, but we drop two classes. In this section, we give experiments and analysis of the reason behind it. We present the quantitative results of two configurations(use or not use data from the two classes for training) in Supp-Tab. 1. Note we are using the same test dataset for the two configurations and the only difference is the amount of training data. *Building&&Ourdoor* is not in the range of the **Object** but various union of different objects. The internal space will bring obstacles for our rendering-based pipeline and some similar pattern in conditional images will also cause ambiguity. *PoorQuality* will influence the manufactured property of our generated shapes and thus make the final geometry looks worse.

Supp-Tab. 1. Quantitative comparison on Objaverse-Clean(***O***) and full Objaverse(***O-full***) when evaluate on the same test split from the clean eight classes.

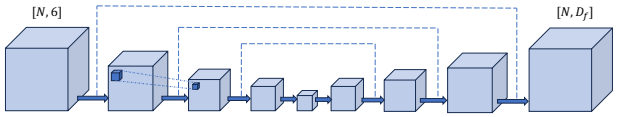
Method	Chamfer Dist.↓ ↑	PSNR↑	SSIM↑	LPIPS↓
Sparse3D (***O-full***)	0.131	18.49	0.887	0.124
Sparse3D (***O***)	**0.108**	**19.23**	**0.908**	**0.110**

4 Rendering Configuration

The main factors we need to consider are two-fold:

- The reprojected point clouds need uniform density everywhere since uneven density will provide the wrong priority for VAE reconstruction. Our VAE learns a mapping function from dense point clouds to coherent textured shapes. If we input point clouds with uneven density, the network needs to learn a function with stronger inductive bias, which makes it more difficult to generalize across classes;
- The renderings need to cover the surface of the mesh as completely as possible. This gives a lower bound for the number of views if the camera's FOV(Field of View) is specific.

To satisfy the above requirements, we render 38 views of a centered object. As depicted in Supp-Fig. 3, we render 24 views at $elevation \in [5^o, 30^o], rotation = \{r \times 15^o | r \in [0, 23]\}$, and 12 views at $elevation \in [-5^o, 5^o], rotation = \{r \times 30^o | r \in [0, 11]\}$, and 2 views for top and bottom respectively.



Supp-Fig. 4. Structure of Sparse Convolution Network.

5 Detailed Structure of Sparse Convolution Network

As depicted in Supp-Fig. 4, we adopt a U-Net-like structure consisting of four downsample blocks and four corresponding upsample blocks. Similar skip connections are adopted to improve the performance. The input is xyz coordinates and corresponding rgb values. At the beginning, the input will be voxelized into dense volume and a hash table will be used to memorize the location of activated coordinates. These coordinates and corresponding colors are fed through the sparse convolution network. The output of SCN is point-wise features with aligned positions, and we set $D_f = 12$ for all experiments. The point-wise features further work as input to the modified VAE of StableDiffusion.

6 More Experimental Results

6.1 Interpolation Result

See Supp-Fig. 6

Supp-Fig. 5. More conditional generation samples on ShapeNetV2.

Supp-Fig. 6. Interpolation results of *Car*, *Chair*, and *Table*. The decoded meshes between two random samples maintain high fidelity and continuity, which shows the smoothness of our neural mapping functions.

6.2 Text-to-3D

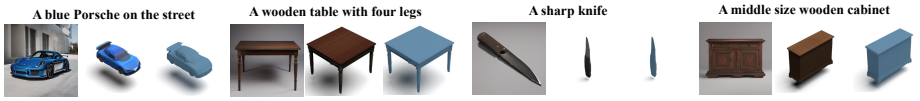
See Supp-Fig. 7

6.3 More conditional results

We have provided some results for benchmark in the paper and we put more conditional results of our method in Supp-Fig. 5. As can be seen, Sparse3D generates high-fidelity textured shapes across various classes and various conditional angles. To avoid cherry-picking of the rendering view, we also visualize multiple views of the generated meshes in Supp-Fig. 8.

6.4 Analysis on loading pre-trained model

Triplanes exhibit spatial correspondence of three orthographic 'images'. As depicted in Supp-Fig. 9, the latent of triplanes(from SparseEncoding) and the latent of 2D images(from StableDiffusion) share similar patterns, motivating us to leverage pre-trained StableDiffusion for domain adaptation and training accelerating. We found that this indeed speeds up the training and costs only $\frac{1}{3}$



Supp-Fig. 7. Visualization of a text-image-3D pipeline utilizing off-the-shell SDXL [34].

Supp-Fig. 8. Multiple views of conditional generation samples on ShapeNetV2.

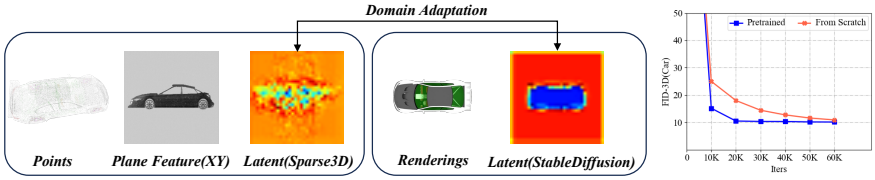
of training time compared with training from scratch. However, the final gain of performance seems relatively subtle(See the **Right** subfigure). Consequently, we have chosen not to emphasize this as a distinct contribution, despite its significant acceleration of the domain adaptation process

6.5 Demo Video

We have prepared a demo video named *uncond.avi* in the zipped files. Please check it for all samples we generate in the unconditional setting. We use Blender for the circular rendering and use only sunlight as the source of light.

6.6 Benchmark on G-Objaverse

Due to the limited computation resources, we only report metrics on the most recent SoTA method. See Supp-Tab. 2 for the qualitative results and we will add



Supp-Fig. 9. (**Left**)latents from Sparse3D and StableDiffusion. (**Right**)Ablation about loading the pre-trained StableDiffusion.

more methods for comparisons in the revision. For the single-class generation, we report metrics of GET3D and Sparse3D on each class of our clean dataset respectively. For the multi-class generation, we report metrics of 3DSVec and Sparse3D on 8 classes. For the label-free generation, we have reported the metrics(Table 3 in the paper), so we abbreviate it here.

Supp-Tab. 2. Quantitative Results on our G-Objaverse.

Settings	Single-Class		Multi-Class	
classes/metrics	FID-3D(GET3D)↓	FID-3D(Sparse3D)↓	Chamfer(3DS2Vec)↓	Chamfer(Sparse3D)↓
Human-Shape	13.92	**11.97**	0.032	**0.027**
Animals	19.68	**13.42**	0.037	**0.030**
Daily-used	47.88	**23.93**	0.041	**0.036**
Furnitures	32.17	**18.34**	0.029	**0.024**
Transportations	19.07	**12.11**	**0.024**	0.025
Plants	57.63	**29.06**	0.058	**0.051**
Food	35.88	**16.13**	0.039	**0.030**
Electronics	52.11	**32.10**	0.048	**0.042**
mean(all)	/	/	0.039	**0.033**

7 TexturedLAS

7.1 Implementation Details

As depicted in Supp-Fig. 10, we add rgb channels in the occupancy diffusion in LASDiffusion, which results in a field diffusion model. This field diffusion model includes an occupancy field and a texture field. Modifications include:

– We use Ball-Query to compute rgb values from dense point clouds when we prepare the ground-truth values of the texture field;

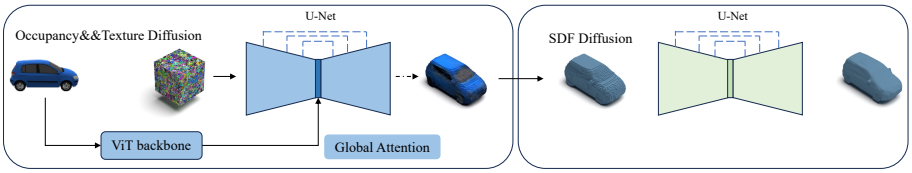
– We use a unified diffusion model to produce the occupancy and the texture field, for which we have modified the input projection layer and the output projection layer;
– We modify the volume resolution from 64 to 128 to further enhance the precision of the texture field and double the base channel of LASDiffusion, which gives a total amount of parameters of 100 MB.

Note we do not modify the second stage since we empirically find that sparse octree-based diffusion models are hard to enhance the texture field.

7.2 Explanation of Black Surface Phenomenon

In Fig. 4, we can easily find that the generated texture of TexturedLAS always contains some black patches. The reasons are two-fold:

– The first stage of TexturedLAS uses a discrete diffusion model for producing better geometry, while it is not the best choice for the continuous texture field in our design;
– When we process the data, the texture field is generated from a specific color band, where all color values are set to be black if the points are not located in this color band. This is inevitable since we must set a specific color of the empty space.



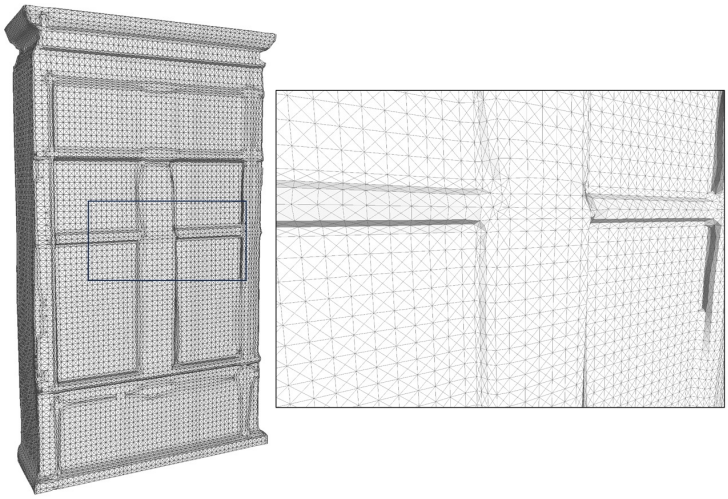
Supp-Fig. 10. Framework of TexturedLAS.

8 Choices of Conditions

Despite achieving some progress in textured shape generation, we find that the single-view conditional paradigm may not be the best choice for the open-vocabulary scenario. Through the experiments in ShapeNetV2(**Aligned**) and Objaverse(**Unaligned**), we find that single-view reconstruction will cause ambiguity if the dataset is not aligned. Here, **Aligned** means that two bananas are posed in the same direction at least in the dataset. If a dataset is not aligned, then there will be two answers for the same conditional image at least, which will further bring obstacles to the network fitting. To solve the ambiguity, there are two ways:

– Align all data;
– Make the network pose-aware.

We need rough class labels and a further definition of the aligned direction if we want to align all data. This requires huge efforts but we have finished the classification. The second way requires us to design a pose-aware generation network and feasible ways may include designing the network to be volume-conditioned and rotating the original 3D data to follow the conditional image, which are future works we are working on.



Supp-Fig. 11. Visualization of the vertices contouring.

9 Manufactured Property

We visualize the vertices contouring of generated samples in Supp-Fig. 11. Currently, the quality of generated 3D content can not satisfy the standard requirements of games, the movie industry, and VR applications, making it important to preserve the manufactured property of meshes for further editing. Our method can preserve the underlying property, which is not ensured by NeRF-based generation methods.