Appendix for Feedback Adversarial Learning: Spatial Feedback for Improving Generative Adversarial Networks

We present additional high resolution results in Section A. We describe the detailed architectures in Section B, followed by the training details presented in Section C.

A. Additional Results

We show additional result on models trained with Feedback for image generation in Figure 1, image-to-image translation in Figure 2 and Figure 3, and voxel generation in Figure 4.

The spatial feedback produced by the discriminators provides useful information for the model to progressively generate better quality images and voxels overtime. In Figure 1, we observe that facial features and the lighting on the skin improve over time. In Figure 2, we can see the texture of the building and street improve with feedback. Similarly Figure 3, we can see that the room lighting improves over time and the colors become more spatially coherent. In Figure 4, the voxel details such as the wings of airplanes, the front side of cars, and the tops of vessels are gradually improved. Also, the noisy voxel cells outside of vehicles are fixed over time.

B. Detailed Architectures

B.1 Image Generation

We use the BigGAN-deep [1] to train our models. We omit using self-attention layer [7] due to the memory overhead. We do not use orthogonal regularization as we have observed the improvement is minor. We use latent vector of size 128 sampled from a truncated normal $z \sim trunc(\mathcal{N}(0, I), 2)$. The latent vector z is passed through the network as the input. The latent vector is also used to predict the affine parameters for the conditional batch normalization layers. See Table 4 for details. The feedback information is fed in after the 4-th residual block – the generator has 10-blocks in total. The construction of the residual blocks can be found in [1].

B.2 Image-to-image Translation

For the task of image-to-image translation, we use the same architecture as [8]. We use 9-Residual blocks. We define the encoding to be at the end of 4-th block. We made some minor modification to improve the overall performance of the original model. We replace strided-convolution with stride-1 convolution followed by average pooling. We replaced the convolution-transpose layer with nearest neighbor upsampling followed by stride-1 convolution. We replaced 4×4 kernels in the discriminator with 3×3-kernels. Lastly, we apply spectral-normalization on all the layers. The composition of the residual block can be found in [8].

B.3 Voxel Generation

We adopt a similar architecture proposed in VoxelGAN [6]. We train the generator to predict $64 \times 64 \times 64$ voxel from a 128-dimensional latent vector. The voxels are constraint to be within $[-1, 1]$. See Table 6 for architecture details.

B.4 Voxel Classifier

The classifier consists of 6 3D-convolutional layers, The last layer is flattened as passed through fully-connected layer to output a 10-dimensional vector indicating the predicted probabilities. See Table 7 for architecture details.

C. Training Details

C.1 Image Generation

We train our models using Adam optimization [4], with $\beta_1 = 0$ and $\beta_2 = 0.999$, and a learning rate of $5 \cdot 10^{-5}$ for the generator and $2 \cdot 10^{-4}$ for the discriminator. We train all our models with a batch size of 16 for 10 epochs.

C.2 Image-to-image Translation

We train our models using Adam optimization [4], with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We use a learning rate of 10^{-4} for both generator and the discriminator. We scale the reconstruction loss by 10. We train all our models with a batch size 1 for 200 epochs.

C.3 Voxel Generation

We train our models with Adam optimization [4] with a learning rate of 10^{-4}, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. We train the models with a batch size of 10. The weight of the gradient penalty (γ) is set to 10 as proposed in [2]. We use a latent dimension of 200, following [6].

Category	Airplane	Bench	Car	Chair	Display
Accuracy	95.9%	77.8%	99.6%	97.3%	95.0%
# of Voxels	4045	1816	7497	4962	1095
Category	Rifle	Speaker	Table	Telephone	Vessel
Accuracy	99.2%	86.9%	95.4%	95.0%	98.8%
# of Voxels	2372	1618	8509	1052	1939

Table 1: **Voxel classifier:** Voxel classifier's classification accuracy. The trained classifier's overall accuracy is 95.9% on ShapeNet.

C.4 CelebA Inception Classifier

To evaluate the quality of the generated CelebA images, we use the Frechet Inception Distance (FID). FID is a widely used metric that uses an inception network trained on ImageNet to measure the statistical discrepancy between the generated samples and real samples. We used the Inception-v3 model provided by PyTorch [5]. We use the last feature layer (dimension of 2048) to compute the FID score.

C.5 Voxel Classifier

To train the voxel classifier, we use all 10 categories in ShapeNet: airplane, bench, car, chair, display, rifle, speaker, table, telephone, and vessel. We collect around 30k training voxels and 6k testing voxels. We train the classifier with a batch size of 32 using Adam optimization [4] with a learning rate of 10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The classifier is trained to perform a 10-way classification using cross-entropy loss. The performance of the classifier in each category is shown in Table. 1. Note that the bench category is original a sub-category of chairs. We separate it from chairs because it has a sufficient number of 3D models. This also explains the lower classification accuracy of benches, which are easily misclassified to chair category.

D. Ablation Study

In Table 2, we experiment with using only previous generation as feedback (similar to [3] from ICLR workshop) which outperforms the baseline without feedback. Our model achieves the best performance using *both the previous generation and the discriminator response* since the model can modify image regions that are considered fake by the discriminator. Furthermore, in Table 3, we experiment with different variations of combining feedback information into the encoded features. We observed predicting both the scale and the bias to achieve the best performance.

Feedback	Per-pixel acc	Class IOU
Ground Truth	0.76	0.21
No feedback	0.70	0.18
Previously Generated Image [3]	0.72	0.18
Discriminator Output	0.71	0.18
Both (ours)	**0.74**	**0.19**

Table 2: **Feedback input** CityScapes FCN score when using different types of feedback (higher is better). Our model uses both the previously generated image and the discriminator response of it as feedback, achieving the best performance.

Generation	$t = 1$	$t = 2$	$t = 3$
Metric	Per-pixel acc		
Ground Truth	0.76	-	-
No feedback	**0.70**	-	-
Bias	0.60	0.70	0.71
Scale	0.58	0.72	**0.74**
AST (ours)	0.61	**0.73**	**0.74**

Table 3: **Variations of feedback** CityScapes segmentation-to-photo performance using different feedback mechanism (higher is better). Bias adds information to the feature, and scale is re-weights the features.

References

[1] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[2] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, 2017.

[3] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. In *ICLR Workshop*, 2016.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[5] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[6] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, 2016.

[7] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *Neural Information Processing Systems*, 2018.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, 2017.

Table 4: Architecture details for unconditional image-generation. We use $ch = 64$.

Generator (Encoder G_e)
$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
Dense $128 \rightarrow 4 \times 4 \times 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $8ch \rightarrow 8ch$
Generator (Decoder G_d)
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock up $2ch \rightarrow ch$
BN, ReLU, 3×3 conv $ch \rightarrow 3$
Tanh

Discriminator (D)
RGB image $y \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down $ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
ResBlock down $4ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, 3×3 Conv $16ch \rightarrow 1$

Feedback (encoder F_a)
Feedback input ($\hat{y}_t$ concat r_t) $\in \mathbb{R}^{128 \times 128 \times 4}$
3×3 Conv $4 \rightarrow ch$, AvgPool, BN, ReLU
3×3 Conv $ch \rightarrow 2ch$, AvgPool, BN, ReLU
3×3 Conv $4ch \rightarrow 4ch$, AvgPool, BN, ReLU
3×3 Conv $4ch \rightarrow 8ch$, AvgPool, BN, ReLU
Feedback (Decoder F_d)
(h_{t-1} concat f_{t-1})
3×3 Conv $16ch \rightarrow 16ch$

Table 5: Architecture details for image-to-image translation. We use $ch = 64$.

Generator (Encoder G_e)
$x \in \mathbb{R}^{H \times W \times 3}$
Conv $3 \rightarrow ch$, AvgPool, IN, ReLU
Conv $ch \rightarrow 2ch$, AvgPool, IN, ReLU
Conv $2ch \rightarrow 4ch$, AvgPool, IN, ReLU
Conv $4ch \rightarrow 8ch$, AvgPool, IN, ReLU
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
Generator (Decoder G_d)
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
NN-upsample, Conv $8ch \rightarrow 4ch$, IN, ReLU
NN-upsample, Conv $4ch \rightarrow 2ch$, IN, ReLU
NN-upsample, Conv $2ch \rightarrow ch$, IN, ReLU
Conv $2ch \rightarrow ch$
Tanh

Discriminator
RGB image $y \in \mathbb{R}^{256 \times 256 \times 3}$
3×3 Conv $3 \rightarrow ch$, AvgPool, IN, LeakyReLU
3×3 Conv $ch \rightarrow 2ch$, AvgPool, IN, LeakyReLU
3×3 Conv $2ch \rightarrow 4ch$, AvgPool, IN, LeakyReLU
3×3 Conv $4ch \rightarrow 8ch$, AvgPool, IN, LeakyReLU
3×3 Conv $8ch \rightarrow 8ch$, AvgPool, IN, LeakyReLU
ReLU, 3×3 Conv 1

Feedback (encoder F_a)
Feedback input ($\hat{y}_t$ ch-concat r_t) $\in \mathbb{R}^{H \times W \times 4}$
3×3 Conv $3 \rightarrow ch$, AvgPool, IN, ReLU
3×3 Conv $ch \rightarrow 2ch$, AvgPool, IN, ReLU
3×3 Conv $2ch \rightarrow 4ch$, AvgPool, IN, ReLU
3×3 Conv $4ch \rightarrow 8ch$, AvgPool, IN, ReLU
ResBlock $8ch \rightarrow 8ch$
Feedback (Decoder F_b)
h_{t-1} ch-concat f_{t-1}
3×3 Conv $16ch \rightarrow 16ch$

Table 6: Architecture details for voxel generation. We use $ch = 32$.

Generator (Encoder G_e)
$z \in \mathbb{R}^{200} \sim \mathcal{N}(0, I)$
Dense 128 $\rightarrow$ 256, LeakyReLU
Dense 256 $\rightarrow 4 \times 4 \times 4 \times ch$, LeakyReLU
Generator (Decoder G_d)
NN-upsample, 3×3 Conv3D $8ch \rightarrow 4ch$, ReLU
NN-upsample, 3×3 Conv3D $4ch \rightarrow 2ch$, ReLU
NN-upsample, 3×3 Conv3D $2ch \rightarrow ch$, ReLU
NN-upsample, 3×3 Conv3D $ch \rightarrow 1$
Sigmoid

Discriminator (D)
Voxel cube $y \in \mathbb{R}^{64 \times 64 \times 64}$
3×3 Conv3D stride-2 $1 \rightarrow 2ch$, LeakyReLU
3×3 Conv3D stride-2 $2ch \rightarrow 4ch$, LeakyReLU
3×3 Conv3D stride-2 $4ch \rightarrow 8ch$, LeakyReLU
3×3 Conv3D stride-2 $8ch \rightarrow 16ch$, LeakyReLU
1×1 Conv3D $16ch \rightarrow 4ch$, LeakyReLU
1×1 Conv3D $4ch \rightarrow 1$

Feedback (encoder F_a)
Feedback input ($\hat{y}_t$ concat r_t) $\in \mathbb{R}^{64 \times 64 \times 64 \times 2}$
3×3 Conv3D stride-2 $2ch \rightarrow 4ch$, BN, ReLU
3×3 Conv3D stride-2 $4ch \rightarrow 8ch$, BN, ReLU
3×3 Conv3D stride-2 $8ch \rightarrow 16ch$, BN, ReLU
3×3 Conv3D stride-2 $16ch \rightarrow ch$, BN, ReLU
3×3 Conv3D $ch \rightarrow ch$, BN
Feedback (Decoder F_b)
(h_{t-1} concat f_{t-1})
3×3 Conv3D $2ch \rightarrow 2ch$

Table 7: Architecture details for voxel classifier. We use $ch = 32$.

Classifier
Voxel cube $y \in \mathbb{R}^{64 \times 64 \times 64}$
3×3 Conv3D stride-2 $1 \rightarrow 2ch$, BN, LeakyReLU
3×3 Conv3D stride-2 $2ch \rightarrow 4ch$, BN, LeakyReLU
3×3 Conv3D stride-2 $4ch \rightarrow 8ch$, BN, LeakyReLU
3×3 Conv3D stride-2 $8ch \rightarrow 16ch$, BN, LeakyReLU
Flatten, Dense $16ch \rightarrow 10$
Softmax

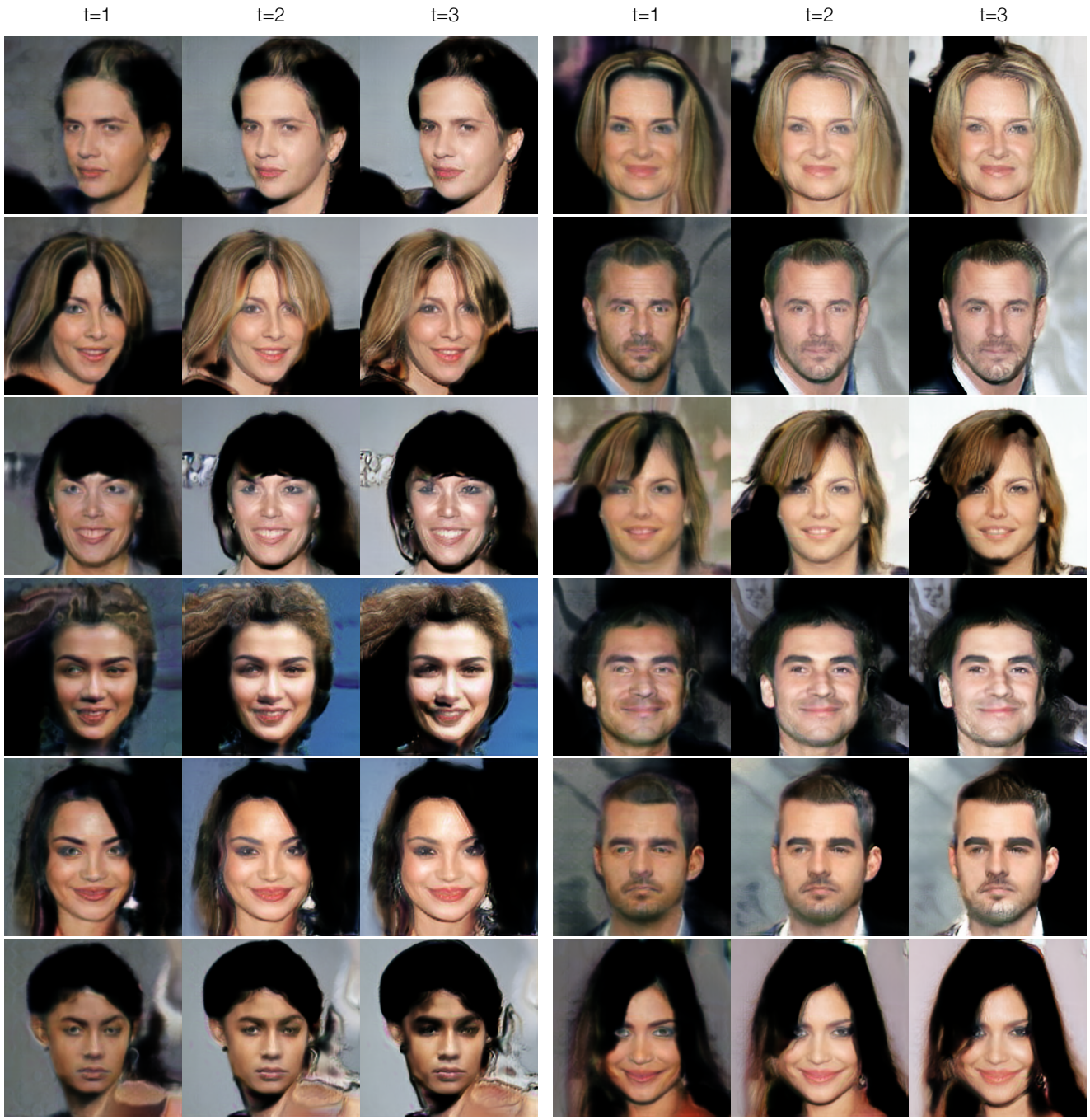

Figure 1: **Image generation**: Additional results on CelebA images.

Figure 2: **Image-to-image translation**: Additional results on Cityscapes images.

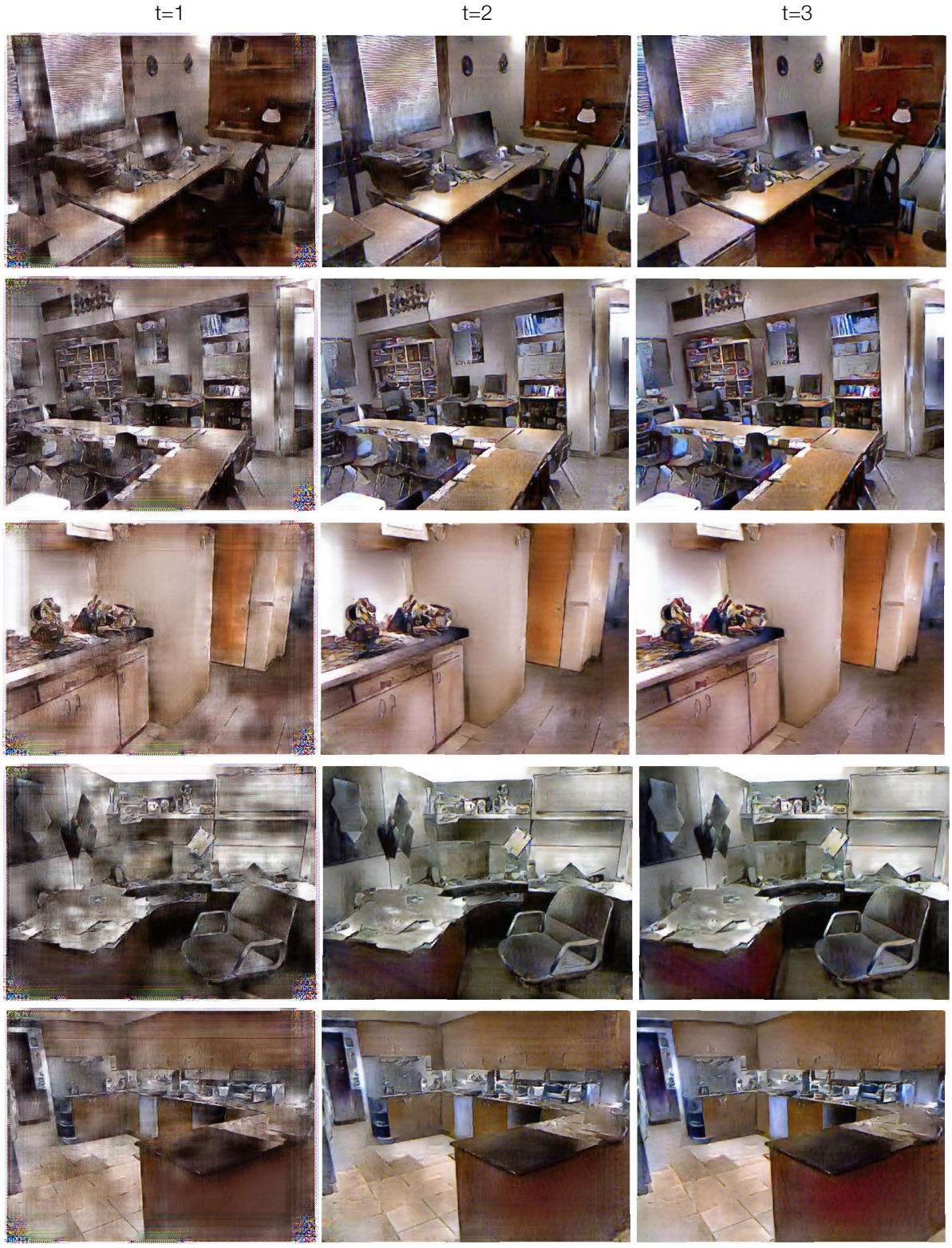

Figure 3: **Image-to-image translation**: Additional results on NYU-depth-v2 images.

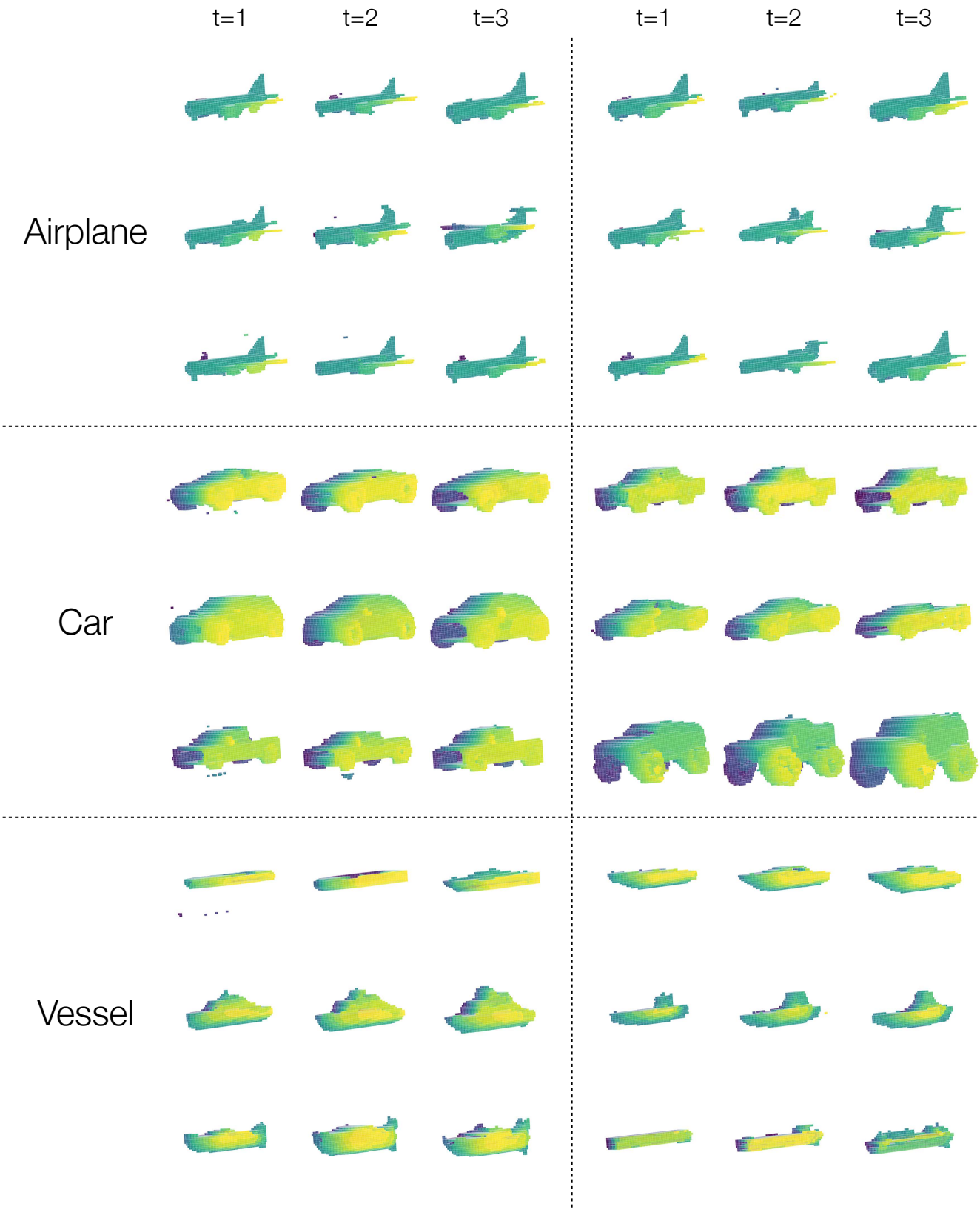

Figure 4: **Voxel generation**: Additional results on ShapeNet voxels.