Monocular 3D Object Reconstruction with GAN Inversion – Supplementary Material –

Junzhe Zhang[1,3], Daxuan Ren[1,3], Zhongang Cai[1,3], Chai Kiat Yeo[2], Bo Dai[4]*, and Chen Change Loy[1]

[1] S-Lab, Nanyang Technological University
[2] Nanyang Technological University
[3] SenseTime Research
[4] Shanghai AI Laboratory
{junzhe001,daxuan001,caiz0023}@e.ntu.edu.sg,
{asckyeo,ccloy}@ntu.edu.sg, {daibo}@pjlab.org.cn

1 Implementation Details

Architectures. We follow the same architectures for the generator and the UV space discriminator as described in ConvMesh [6] for pre-training. ConvMesh baseline uses a convolutional generator G with two branches, to generate deformation map $\mathbf{S} \in \mathbb{R}^{32\times32}$ and texture map $\mathbf{T} \in \mathbb{R}^{512\times512}$ in the UV space respectively from a latent code $\mathbf{z} \in \mathbb{R}^{64}$. The UV space discriminator consists of two sub-discriminators, for discriminating the deformation map and texture map respectively. In addition, we introduce an image space discriminator to further enforce the realism of the synthesized texture and shape, following the architecture of PatchGAN [3]. We render the synthesized textured mesh to images with the DIB-R differentiable renderer [1] following the Kaolin [4] implementation.
Preparation of Pseudo Ground Truths. During pre-training, discrimination in the UV space requires pseudo ground truth deformation maps and texture maps of the training images. The pseudo deformation maps are obtained by training a mesh reconstruction baseline on the training set. It outputs a 3D object as a deformation map and a texture map as well. Subsequently, the associated pseudo texture maps are obtained through a form of inverse rendering, *i.e.*, projecting the foreground pixels from the natural images onto the UV space. Since only the visible regions can be projected to the UV space, all the resulting pseudo texture maps are in fact partial. During training, the generated texture maps are masked to form partial texture maps as well prior to discrimination. More details of pseudo ground truths preparation can be found in [6].

Note that the mesh reconstruction baseline model is overfitted for shape estimation to the training set and is not suitable for 3D reconstruction. In particular, the resulting network generally gives blurry textures, and the shape estimation does not generalize very well to unseen images. Quantitatively, it gives an IoU of 0.671 in contrast to our method with an IoU of 0.752.

* Bo Dai completed this work when he was with S-Lab, NTU.

Pre-training. We pre-train ConvMesh following a class conditional setting for 600 epochs, with a batch size of 128, which takes 15 hours on four Nvidia V100 GPUs. The generator is updated once every three iterations with a learning rate of 1×10^{-4} whereas the discriminators are updated concurrently twice every three iterations with a learning rate of 4×10^{-4}. We use the Adam optimizer [5] with $\beta_1 = 0$ and $\beta_2 = 0.9$. For the objective function $\mathcal{L}_G$ in the main paper, we let $\lambda_{uv} = 1$ and $\lambda_I = 0.04$. We use the same settings for CUB and PASCAL 3D+.

Thanks to the discriminator in the image space, our pre-training results are better than the baseline with a clear margin, as shown in Tab. 1.

Table 1. Pre-training results on CUB comparing ConvMesh baseline and ours with the image space discrimination. The Full FID is computed on generated meshes and generated textures; the Texture FID is computed on the generated texture and mesh estimated using the differentiable renderer; the Mesh FID is computed on the pseudo ground truth texture with predicted mesh. We report FID with truncated $\sigma = 0.25$.

	Full	Texture	Mesh
ConvMesh baseline	33.6	28.7	19.5
Ours w/ image space discrimination	**28.3**	**27.2**	**18.7**

Inversion. We adapt a multi-stage inversion with different learning rates, with learning rates of the latent code $[1 \times 10^{-1}, 5 \times 10^{-2}, 1 \times 10^{-2}, 5 \times 10^{-3}]$, learning rates of the camera pose $[1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}]$, and iterations $[50, 50, 50, 50]$. We use the Adam optimizer with $\beta_1 = 0$ and $\beta_2 = 0.99$. For each testing instance, the inversion process takes around 40 seconds, and our framework supports distributed inference. For the overall inversion loss $\mathcal{L}_{inv}$, we set the weights of $\mathcal{L}_{pCT}$, $\mathcal{L}_{fCT}$, $\mathcal{L}_{CM}$, $\mathcal{L}_{smooth}$ and $\mathcal{L}_z$ as 1, 0.05, 10, 0.00005 and 0.05 respectively. For the Chamfer texture losses, we let $\epsilon_s = 0.9$, $\epsilon_a = 1$, and $\alpha = 1$. In particular, $1 - \epsilon_s$ corresponds to degree of tolerance to local misalignment. We show in Tab. 2 that a smaller ϵ_s gives relatively better results in the presence of imperfect camera poses and high-frequency details. One can tighten ϵ_s if more accurate camera poses are given.

Table 2. Effect of ϵ_s in Chamfer texture losses.

ϵ_s	IoU ↑	FID_1 ↓	FID_{10} ↓	FID_{12} ↓
0.999	0.747	38.7	39.8	58.0
0.99	0.746	38.6	40.0	58.3
0.98	0.748	38.7	39.7	57.8
0.95	0.749	37.6	39.0	57.2
0.9	**0.752**	**37.3**	**38.7**	**56.8**

Test-time Optimization of Baselines. Similar to GAN inversion, a relatively compact latent space is desirable for efficient optimization during the test time. Both CMR and UMR have a latent code with a dimension of 200. In contrast, U-CMR has a latent code with a dimension of 4096, whereas SMR does not follow an auto-encoder architecture, but directly encodes the 3D attributes from the image with the associated mask. Therefore, SMR and U-CMR are infeasible to be adapted for test-time optimization.

We adapt CMR and UMR as follows during the test time: The latent code is first obtained with a single forward pass by the image encoder, and the camera pose is then obtained by the camera pose estimator. We then fine-tune the latent code and the camera pose by minimizing the mask loss and texture loss by comparing against the estimated mask and the input image respectively, where the network weights remain fixed. The choices of loss functions and their weights follow those during the training time. For an equal comparison, we fine-tune with the same Adam optimizer and for the same number of iterations, 200, for each testing instance. Since our method uses randomly initialized latent code whereas the forward pass by the image encoder already provides a good initialization, we use a smaller learning rate for the latent code, 5×10^{-3}. For the camera pose, we use the same learning rates as our method following the multi-stage setting.
Additional Details for User Study. We conduct a user preference study on CUB to evaluate our method. This user preference study involves 40 users, 30 objects, and five methods (four baselines and ours). The 40 users are invited from several different backgrounds, including finance, business, life science, and information technology. We randomly choose 30 objects from the testing split, and ensure the following varieties are contained in the selection: complex and a wide range of texture, with highly articulated shapes, and in the presence of occlusion, etc. The reconstructed 3D objects are rendered from three different viewpoints to make sure that the entire object is observable by the user. For each input image, we give users unlimited time to select the method that gives the most faithful and realistic result in terms of three separate criteria: texture quality, shape quality, and overall textured shape reconstruction.

2 Comparison with Shelf-Sup

Unlike mainstream approaches compared in the main paper that directly deform a category-level template mesh, Shelf-Sup [2] first gives a coarse volumetric prediction, and then converts the coarse volume into a mesh followed by test-time optimization. This design demonstrates its scalability on significantly more categories with various topologies (though still category-specific). However, such scalability is at the sacrifice of categorical semantic consistency without a mesh template, leading to less compelling reconstructions on birds, as compared in Tab. 3. Both Shelf-Sup and UMR employ adversarial loss when training the auto-encoders for reconstruction, whereas the generative adversarial training in GANs for 3D object synthesis offers richer prior, giving better generalization and photo-realism as demonstrated in Sec 4.1. In addition, Shelf-Sup also tends

to suffer from blurry reconstructions with imperfect camera predictions, which calls for the robustness offered by Chamfer texture loss.

Table 3. Comparison with Shelf-Sup on CUB.

Methods	IoU ↑	FID_1 ↓	FID_{10} ↓	FID_{12} ↓
Shel-Sup	0.707	81.2	140.1	161.0
Ours	**0.752**	**37.3**	**38.7**	**56.8**

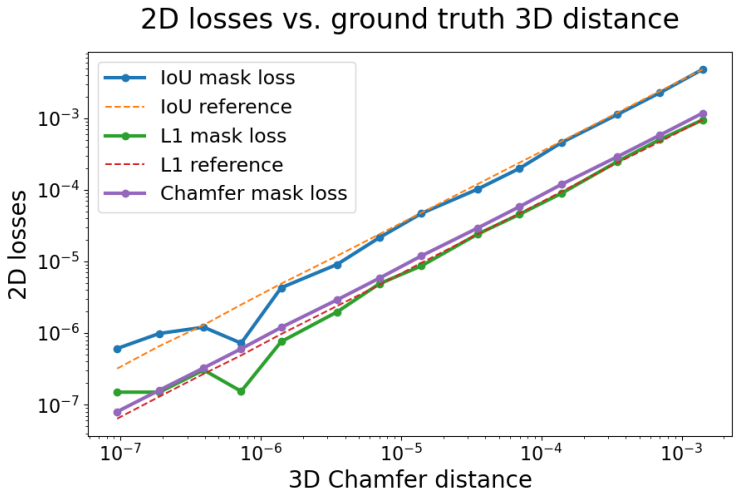

Fig. 1. Sensitivity comparison of various mask losses. Unlike existing mask losses, the constant slope by Chamfer mask loss implies its high accuracy across a wide range of shape variation. Each data point is the mean of 100 simulation runs.

3 Sensitivity Comparison of Mask Losses

We quantitatively analyze the sensitivity of various mask losses by measuring the distance between two 3D shapes at different degrees of shape variations. Specifically, we utilize the pre-trained ConvMesh to randomly generate 100 3D shapes. For each shape O_i, we introduce a variation of the shape by jittering its latent code z_i by a step size η at a random direction, giving the deviated shape O_i'. We then measure the ground truth distance in the 3D space between O_i and O_i' using Chamfer distance, and compute the distances in the 2D space using IoU loss, L1 loss, and Chamfer mask loss respectively. Specifically, we vary the step size η from 1×10^{-6} all the way to 1×10^{-1}, giving 3D Chamfer distances from 1×10^{-7} to 1×10^{-3}. Given these generated and jittered 3D shape pairs, we plot various 2D mask losses against ground truth 3D Chamfer distance in Fig. 1.

As all the four metrics, including Chamfer distances, are L1-like, all the three 2D losses should be linearly correlated to the 3D Chamfer distance, *i.e.*, these plots should maintain constant slopes. However, due to discretization-induced information loss, both IoU mask loss and L1 mask loss are not able to accurately reflect the subtle variation in the 3D shape. Such inaccuracy eventually leads to insensitive gradients and undermines geometry recovery. In contrast, $\mathcal{L}_{CM}$ intercepts the rasterization process and naturally retains information.

4 Qualitative Results for Texture Loss Ablation

We provide qualitative results for ablation study on texture losses in Fig. 2 (zoom in for details). As inaccurate camera initialization easily leads to reprojection misalignment, in the presence of high-frequency texture, conventional pixel-aligned appearance losses, *e.g.*, L1 loss on RGB images and perceptual loss on feature maps, would tend to give noisy supervision signal. This often leads to blurry reconstructions, especially for L1 loss. While earlier attempt to address the misalignment issue by contextual loss treats feature maps as sets of feature vectors, it totally ignores the feature locations in the image coordinate, resulting in unfaithful reconstructions.

Fig. 2. Ablation study on Chamfer texture losses.

5 Additional Qualitative Results

We provide more single-view illustrative examples for CUB in Fig. 3 and Fig. 4. In addition, we further demonstrate the merit of our method through 360-degree comparisons with baselines in the supplemental video. The supplemental video also includes illustration of the inversion process, and 360-degree results for texture transfer and on PASCAL3D+ Car.

Fig. 3. Additional qualitative results on CUB.

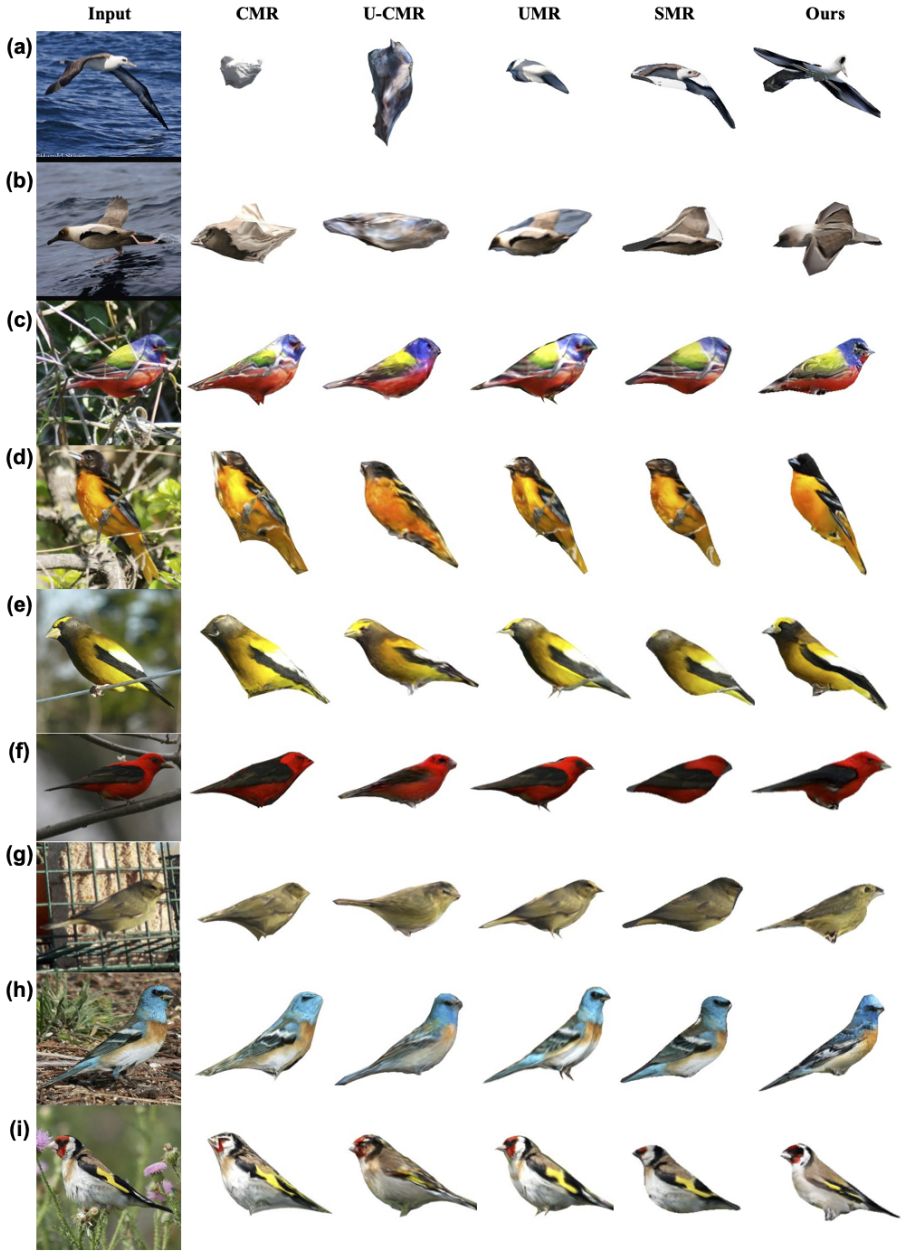

Fig. 4. Additional qualitative results on CUB (continued).

References

1. Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to predict 3D objects with an interpolation-based differentiable renderer. In: NeurIPS (2019) 1
2. *et al*, Y.: Shelf-supervised mesh prediction in the wild. In: CVPR (2021) 3
3. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017) 1
4. Jatavallabhula, K.M., Smith, E., Lafleche, J.F., Tsang, C.F., Rozantsev, A., Chen, W., Xiang, T., Lebaredian, R., Fidler, S.: Kaolin: A PyTorch library for accelerating 3D deep learning research. CoRR **abs/1911.05063** (2019) 1
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) 2
6. Pavllo, D., Spinks, G., Hofmann, T., Moens, M.F., Lucchi, A.: Convolutional generation of textured 3D meshes. In: NeurIPS (2020) 1