MESHINVERSION: 3D TEXTURED MESH RECONSTRUCTION WITH GENERATIVE PRIOR – SUPPLEMENTARY MATERIAL –

Anonymous authors
Paper under double-blind review

1 ADDITIONAL DETAILS FOR PRE-TRAINING

We follow the same architectures for the generator and the UV space discriminator as described in Pavllo et al. (2020) for pre-training. ConvMesh baseline (Pavllo et al., 2020) 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 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 (Isola et al., 2017). We render the synthesized textured mesh to images with a differentiable renderer (Chen et al., 2019) following the Jatavallabhula et al. (2019) implementation.

Preparation of the UV space pseudo ground truths, which are processed from the training images and used for UV space discrimination, requires to first train a reconstruction network similarly to training CMR. With this purpose, it is overfitted to the training set for shape estimation, and not desirable for 3D reconstruction. Specifically, the resulting network generally gives blurry textures, and the shape estimation does not generalize very well to unseen images. Quantitatively, the encoder/decoder gives IoU of 0.671 in contrast to MeshInversion with IoU of 0.708. With the trained auto-encoder, pseudo ground truths for the deformation maps can be extracted, and pseudo ground truths for the texture maps are prepared by a form of inverse rendering, more details of which can be found in Pavllo et al. (2020).

For the objective function $\mathcal{L}_G$ in the main paper, we let $\lambda_{uv} = 1$ and $\lambda_I = 0.04$. We pre-train ConvMesh following a class conditional setting on four Nvidia V100 GPUs for 600 epochs, with a batch size of 128. 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 same settings for CUB and PASCAL3D+.

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	33.63	28.68	19.49
Ours w/ 2D domain discrimination	**28.29**	**27.16**	**18.70**

2 ADDITIONAL DETAILS FOR INVERSION

We report the hyperparameters used in the GAN inversion stage. For the proposed Chamfer Texture loss, we let $\epsilon_s = 0.98$, $\epsilon_a = 1$, and $\alpha = 1$. For the weights of various losses, we let $\lambda_{CT} = 1$, $\lambda_{CM} = 3$, $\lambda_{smooth} = 0.00005$, and $\lambda_z = 0.05$. We adapt a multi-stage inversion with different

Table 2: Ablation study. Our proposed Chamfer texture losses $\mathcal{L}_{CT-p}$ and $\mathcal{L}_{CT-f}$ and Chamfer mask loss $\mathcal{L}_{CM}$ are effective to address the misalignment and quantization challenges induced by rendering.

Mask loss	Texture loss	IoU ↑	FID_1 ↓	FID_{10} ↓	FID_{12} ↓
IoU loss	L1 + perceptual loss	0.588	62.3	60.6	76.3
L_{CM}	L1 loss	0.705	71.2	97.3	108.4
	L2 loss (MSE)	**0.708**	75.2	108.1	117.2
	perceptual loss	0.701	49.8	52.3	69.5
	contextual loss	0.699	65.2	72.5	81.1
	L1 + perceptual loss	0.707	47.1	50.8	66.7
L1 loss	L_{CT-p} + L_{CT-f}	0.582	51.7	53.2	64.4
IoU loss	L_{CT-p} + L_{CT-f}	0.605	51.2	50.8	62.0
L_{CM}	L_{CT-p}	**0.708**	47.9	45.4	63.7
L_{CM}	L_{CT-p} + L_{CT-f}	**0.708**	**38.6**	**38.6**	**56.6**

learning rates, with learning rates of the latent code $[1 \times 10^{-1}, 5 \times 10^{-2}, 1 \times 10^{-2}, 5 \times 10^{-3}]$ and iterations $[50, 50, 50, 50]$. We use the Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0$ and $\beta_2 = 0.99$.

3 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.

4 Extended Ablation Study

We provide a more comprehensive ablation studies on mask and texture losses for our MeshInversion framework in Tab. 2. This table includes the ablation study of the main paper, and also includes a version of our method without neither the Chamfer mask loss nor the Chamfer texture loss, and a version with only pixel-level Chamfer texture loss.

While we base our experiments mainly on silhouette ground truths and cameras *estimated* by structure-from-motion (SfM), we also validate the robustness of our method under relaxed conditions with less perfect camera poses and masks. Note that the silhouette mask can be estimated by off-the-shelf instance segmentation methods; the involvement of a pre-trained GAN in our framework simplifies the task and allows us to train a camera pose estimator individually. Since instance segmentation and camera pose estimation are not the focus of this study, we evaluate under camera poses predicted by an off-the-shelf camera pose estimator from Kanazawa et al. (2018), and under masks predicted by PointRend Kirillov et al. (2020), which is pre-trained on COCO (Lin et al., 2014) without fine-tuning. The predicted cameras by CMR gives 6.03 degree of azimuth error and 4.33 degree of elevation error. The predicted masks by PointRend give an IoU of 0.886. The results show that **our method is reasonably robust to inaccurately predicted camera poses and invariant to masks predicted by off-the-shelf instance segmentation methods**.

We report exact values of 2D mask distances and 3D Chamfer distances The are plotted in Fig. 1.

Table 3: Robustness study of MeshInversion under predicted camera pose predictions and segmentation masks. The results show that our method is reasonably robust to inaccurately predicted camera poses and invariant to masks predicted by off-the-shelf instance segmentation methods.

Condition	IoU ↑	FID_1 ↓	FID_{10} ↓	FID_{12} ↓
Predicted cameras by CMR	0.703	43.1	44.1	59.9
Predicted masks by PointRend	**0.710**	**37.9**	38.9	56.7
Cameras by SfM and ground-truth masks	0.708	38.6	**38.6**	**56.6**

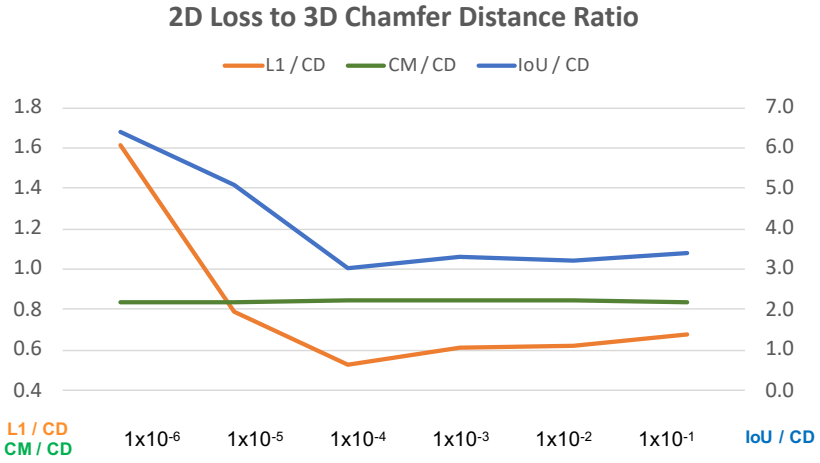

Figure 1: We plot 2D distance to 3D CD ratios between a randomly generated shape by pre-trained ConvMesh and its variation. x-axis is the step size which corresponds to the degree of variation. Note that we take L1 form for 3D Chamfer distance and Chamfer mask loss. Therefore, a varying ratio across shape variation implies inaccuracy in the 2D losses, due to discretization during rasterization. We report exact values of the distances in Tab. 4.

5 Analysis of Various Mask Losses

In this section, we quantitatively analyze the accuracy 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 deviating 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 between O_i and O_i' in the 3D space using Chamfer distance, and compute the distances in the 2D space using IoU loss, L1 loss, and Chamfer mask loss respectively. Note that we take the L1 form for Chamfer distance and Chamfer mask loss. Therefore, all these L1-like 2D distances should ideally be linearly correlated to the 3D Chamfer distance.

We report the average 2D distances and 3D ground truth distance in Tab. 4, and We tabulate these distances in and plot the mask losses to 3D Chamfer distance ratio in Fig. 1 under different step size values from 1×10^{-6} all the way to 1×10^{-1}, where a larger step size corresponds to a larger shape variation, *i.e.*, larger distance between two shapes.

As a result of quantization during the rasterization process, we can obverse from Fig. 1 that both IoU loss and L1 loss have a varying ratio to the ground truth distance at small shape variations. This is particularly harmful to a well-trained ConvMesh, as can be seen from Tab. 4, a small perturbation in the latent space usually corresponds to a slight variation in the 3D shape, in which discretization-induced loss error might be detrimental for geometric learning. In contrast, Chamfer mask loss intercepts the rasterization process to retain information, giving a consistent ratio with respect to the 3D Chamfer distance throughout a wide range of shape variations.

Table 4: We report exact values of 2D mask distances and 3D Chamfer distances between a randomly generated shape by pre-trained ConvMesh and its variation. Step size corresponds to the degree of variation. We take L1 form for 3D Chamfer distance and Chamfer mask loss. The 2D distance to 3D CD ratios are plotted in Fig. 1.

step size η	IoU mask loss	L1 mask loss	Chamfer mask loss	3D Chamfer distance
1.0E-06	6.1E-07	1.5E-07	8.0E-08	9.5E-08
1.0E-05	9.9E-07	1.5E-07	1.6E-07	1.9E-07
1.0E-04	4.3E-06	7.6E-07	1.2E-06	1.4E-06
1.0E-03	4.7E-05	8.7E-06	1.2E-05	1.4E-05
1.0E-02	4.6E-04	0.9E-04	1.2E-04	1.4E-04
1.0E-01	4.8E-03	0.9E-03	1.2E-03	1.4E-03

Table 5: Quantitative results on CUB for test-time optimization (TTO) of baseline methods. With TTO, existing baselines overall achieve a higher fidelity, but our method still remains highly competitive with a clear margin.

	IoU ↑	FID_1 ↓	FID_{12} ↓	FID_{10} ↓
CMR baseline	0.703	140.9	176.2	180.1
CMR + TTO	0.720	122.5	153.26	159.5
UMR baseline	0.734	40.0	72.8	86.9
UMR + TTO	**0.742**	38.7	78.9	90.2
ours	0.708	**38.6**	**38.6**	**56.6**

6 Comparison with Test-time Optimization of Baselines

As our proposed method is essentially test-time optimization of a pre-trained GAN, we also conduct test-time optimization of pre-trained auto-encoders on top of baseline methods for a fair comparison. 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 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. We then fine-tune the embedded feature extracted from the image encoder by minimizing the mask loss and texture loss by comparing against the mask and the 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 MeshInversion uses randomly initialized latent code whereas the forward pass by the image encoder already provides a good initialization, we use a smaller learning rate, 1×10^{-4}.

As the results are shown in Tab. 5, test-time optimization overall gives a higher fidelity for baseline methods, but our method remains superior by a clear margin. Interestingly, UMR with test-time optimization achieves limited improvement in terms of IoU and single-view FID at the cost of worsening novel-view FID. This fair comparison further shows the superiority of generative prior captured through adversarial training over that captured in the auto-encoder.

7 Additional Qualitative Results

We provide more single-view illustrative examples for birds in Fig. 2 and Fig. 3, and more multi-view results for birds in Fig. 4, and for cars in Fig. 5. These extensive examples demonstrate show the good quality of our 3D reconstructions.

REFERENCES

Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. Advances in Neural Information Processing Systems, 32:9609–9619, 2019.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134, 2017.

Krishna Murthy Jatavallabhula, Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. arXiv:1911.05063, 2019.

Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 371–386, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9799–9808, 2020.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In European conference on computer vision, pp. 740–755. Springer, 2014.

Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. Convolutional generation of textured 3d meshes. arXiv preprint arXiv:2006.07660, 2020.

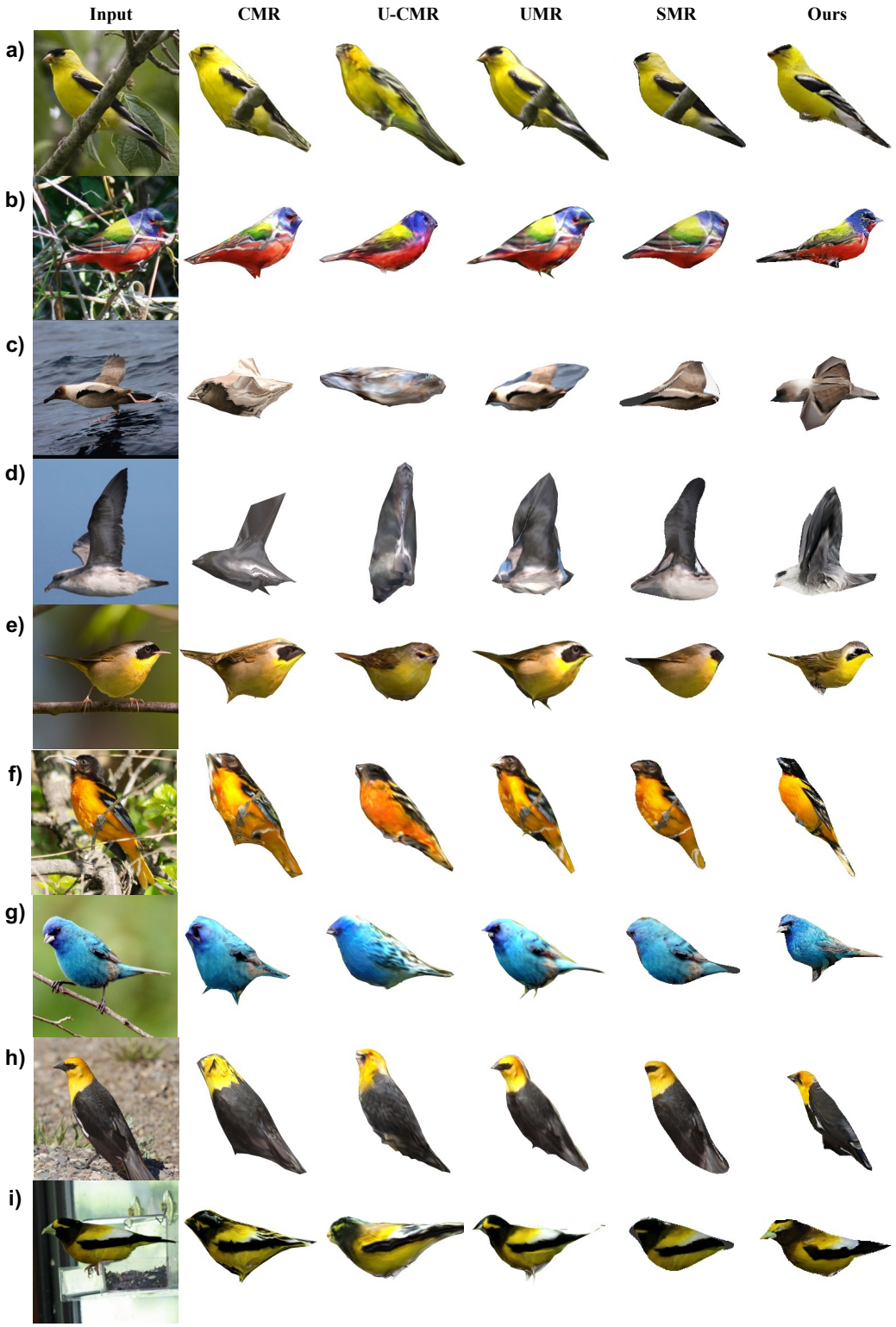

Figure 2: Additional qualitative results on CUB.

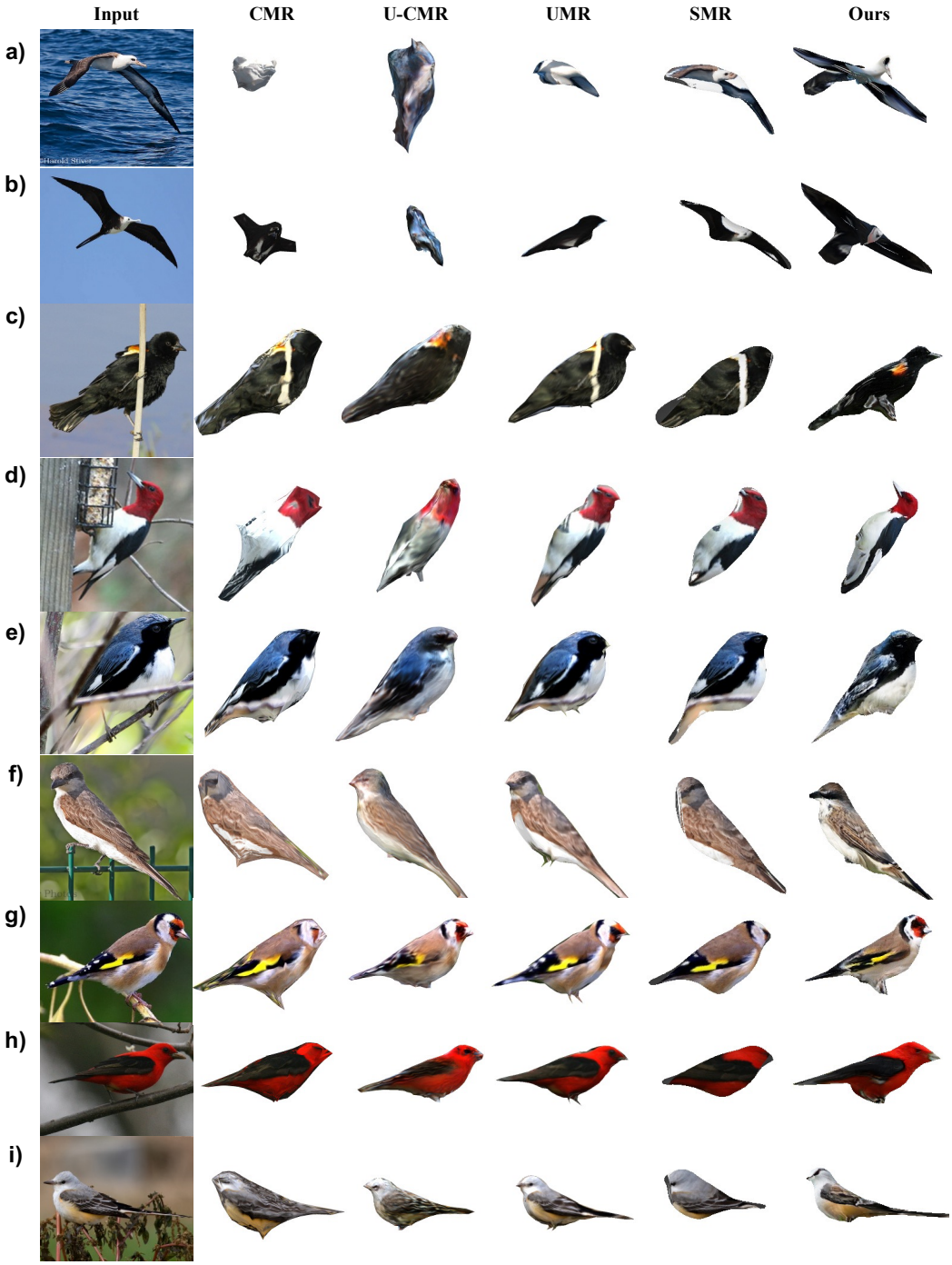

Figure 3: Additional qualitative results on CUB (continued).

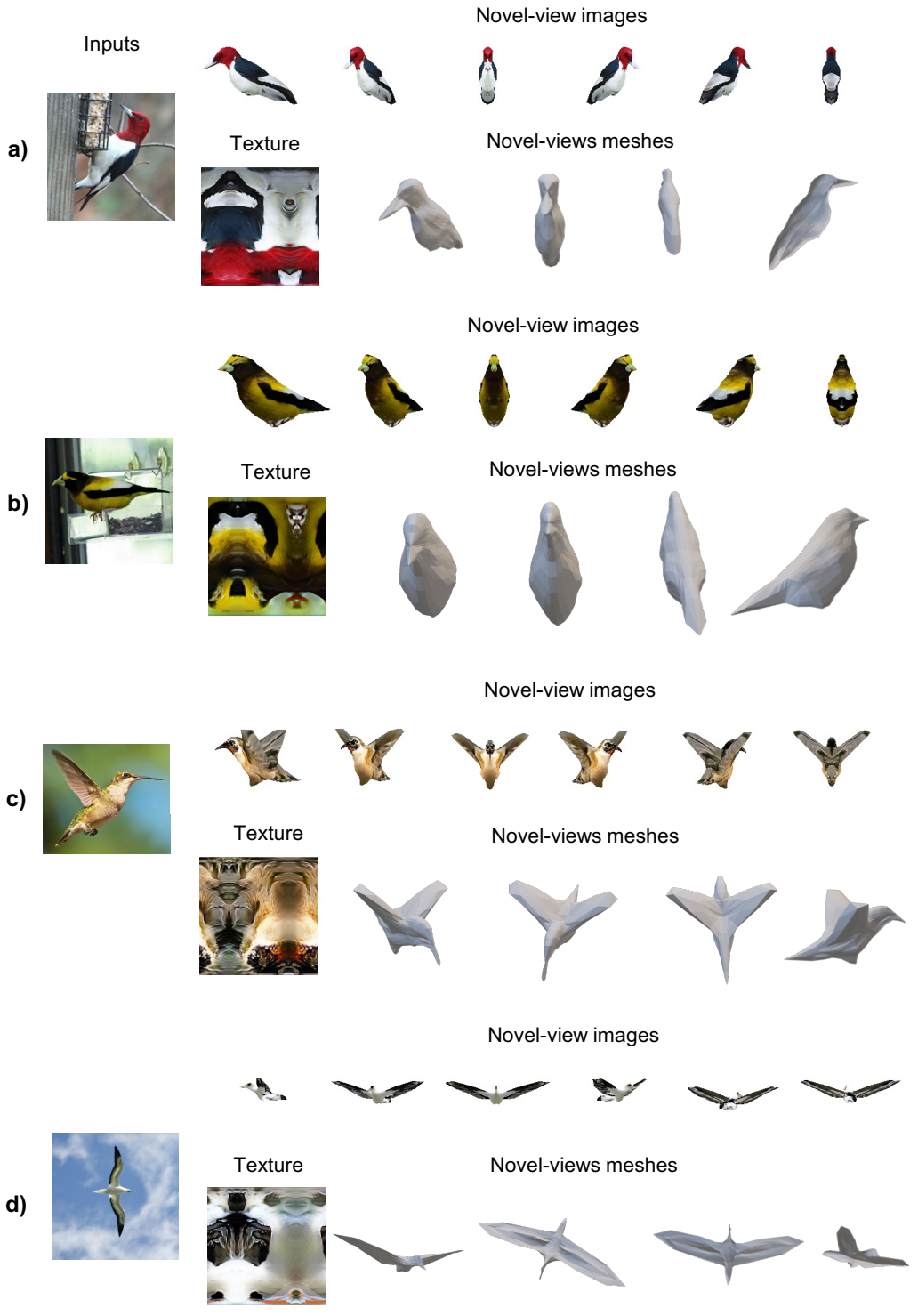

Figure 4: More novel-view qualitative results for CUB. We present both rendered images and meshes from different views, and the texture map as well. Our method achieves realistic 3D reconstruction even for challenging articulations, *e.g.*, birds with open wings. Note that the meshes rendered in Microsoft Powerpoint is perspective, whereas the images rendered with our differentiable renderer is weak perspective.

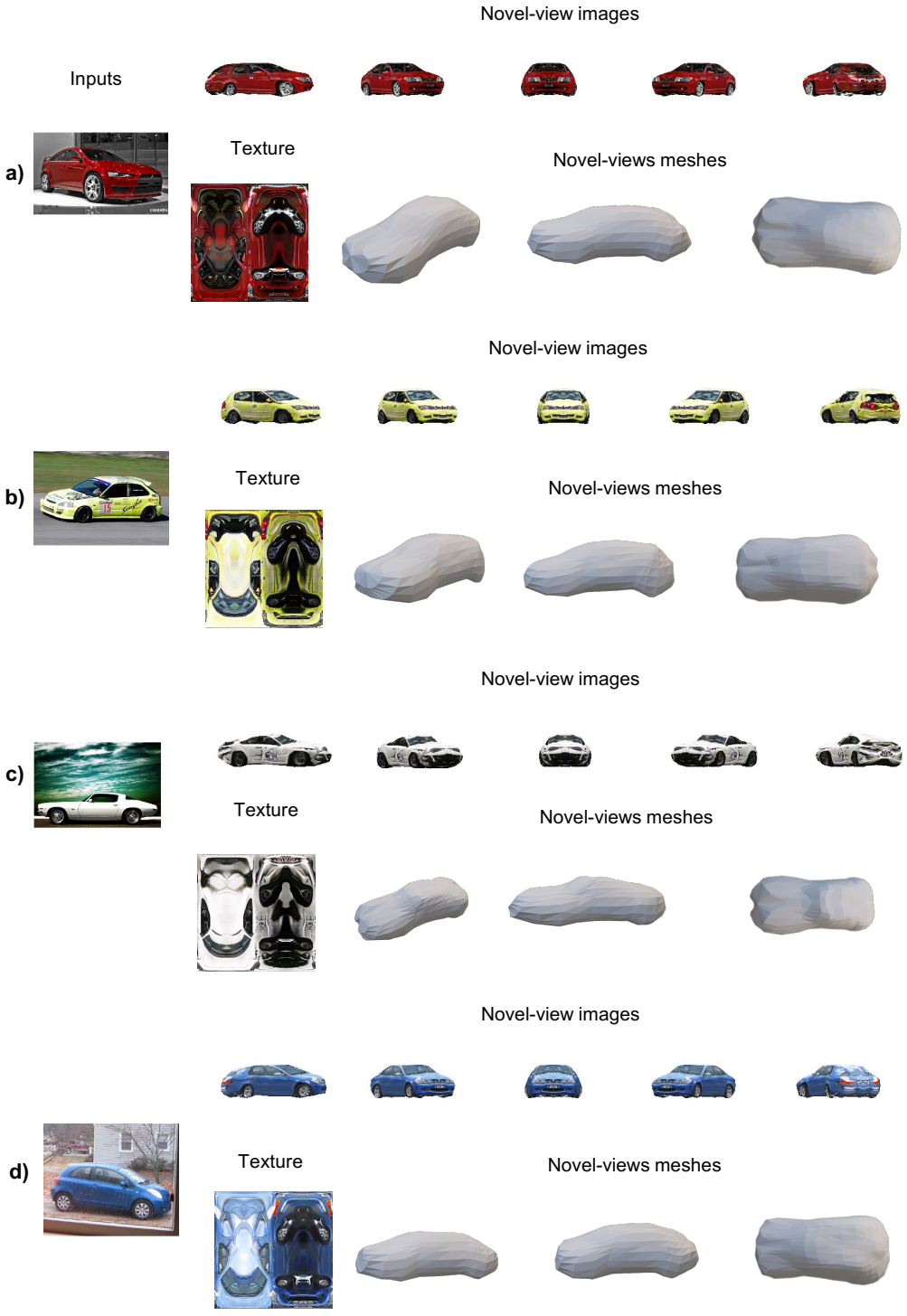

Figure 5: More novel view results for PASCAL3D+ Car.