Supplementary Material for Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency

Tom Monnier[1] Matthew Fisher[2] Alexei A. Efros[3] Mathieu Aubry[1]

[1]Ecole des Ponts, Univ Gustave Eiffel, LIGM [2]Adobe Research [3]UC Berkeley

Abstract. In this supplementary document, we first describe our custom differentiable rendering function (Section 1). Then, we present additional model insights (Section 2) as well as quantitative evaluation details (Section 3). Finally, we provide implementation details (Section 4), including network architectures, design choices and training details.

1 Custom differentiable rendering

Our output images correspond to the soft rasterization of a textured mesh on top of a background image. We observe divergence results when learning geometry from raw photometric comparison with the standard SoftRasterizer [11] and thus propose two key changes. In the following, given a mesh $\mathbb{M}$ and a background $\mathbf{B}$, we describe our rendering function $\mathcal{R}$ producing the image $\hat{\mathbf{I}} = \mathcal{R}(\mathbb{M}, \mathbf{B})$. We first present SoftRasterizer formulation and its limitations, then introduce our modifications. In the following, we write pixel-wise multiplication with $\odot$ and the division of image-sized tensors corresponds to pixel-wise division.

SoftRasterizer formulation. Given a 2D pixel location i, the influence of a face j is modeled by an occupancy function:

$$\mathcal{O}_{\text{SR}}(i,j) = \text{sigmoid}\left(\frac{\nu(i,j)}{\sigma}\right), \tag{1}$$

where σ is a temperature, $\nu(i,j)$ is the signed Euclidean distance between pixel i and projected face j. Let us call $L-1$ the maximum number of faces intersecting the ray associated to a pixel and sort, for each pixel, the intersecting faces by increasing depth. Image-sized maps for occupancy $\mathbf{O}_\ell$, color $\mathbf{C}_\ell$ and depth $\mathbf{D}_\ell$ are built associating to each pixel the ℓ-th intersecting face attributes. Background is modeled as additional maps such that $\mathbf{O}_L = 1, \mathbf{C}_L = \mathbf{B}$, and $\mathbf{D}_L = d_{\text{bg}}$ is a constant, far from the camera. The SoftRasterizer's aggregation function $\mathcal{C}_{\text{SR}}$ merges them to render the final image $\hat{\mathbf{I}}$:

$$\mathcal{C}_{\text{SR}}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}, \mathbf{D}_{1:L}) = \sum_{\ell=1}^{L} \frac{\mathbf{O}_\ell \odot \exp(\mathbf{D}'_\ell/\gamma)}{\sum_k \mathbf{O}_k \odot \exp(\mathbf{D}'_k/\gamma)} \odot \mathbf{C}_\ell, \tag{2}$$

where γ is a temperature parameter, $\mathbf{D}'_\ell = \frac{d_{\text{far}} - \mathbf{D}_\ell}{d_{\text{far}} - d_{\text{near}}}$ and $d_{\text{near}}, d_{\text{far}}$ correspond to near/far cut-off distances. This formulation hence relies on 5 hyperparameters

(σ, γ, d_{near}, d_{far}, d_{bg}) and default values are $\sigma = \gamma = 10^{-4}$, $d_{\text{near}} = 1$, $d_{\text{far}} = 100$ and $\frac{d_{\text{far}} - d_{\text{bg}}}{d_{\text{far}} - d_{\text{near}}} = \epsilon = 10^{-3}$.

The formulation introduced in Equation (2) has one main limitation: gradients don't flow well through $\mathbf{O}_{1:L}$ obtained by soft rasterization, and thus vertex positions cannot be optimized by raw photometric comparison. The simple case of a single face on a black background gives:

$$\hat{\mathbf{I}} = \frac{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}}{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma} + e^{\epsilon/\gamma}} \odot \mathbf{C}_1 \approx \frac{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}}{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}} \odot \mathbf{C}_1 = \mathbf{C}_1, \tag{3}$$

for almost all $\mathbf{O}_1, \mathbf{D}'_1$. Indeed, considering $x, \eta > 0$, we have $xe^{(\epsilon+\eta)/\gamma} \gg e^{\epsilon/\gamma}$ iif $x \gg e^{-\eta/\gamma}$. Even in the best case where $\eta = \epsilon = 10^{-3}$ (*i.e.*, the object is close to d_{far}), this holds for all $x \gg e^{-10} \approx 4 \times 10^{-5}$! We found that tuning γ was not sufficient to mitigate the issue, one would have to tune $\gamma, d_{\text{near}}, d_{\text{far}}, d_{\text{bg}}$ simultaneously to enable the optimization of the vertex positions.

Our layered formulation. Inspired by layered image models [9,14], we propose to model the rendering of a mesh as the layered composition of its projected face attributes. More specifically, given occupancy $\mathbf{O}_{1:L}$ and color $\mathbf{C}_{1:L}$ maps, we render an image $\hat{\mathbf{I}}$ through the classical recursive alpha compositing:

$$\mathcal{C}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}) = \sum_{\ell=1}^{L} \Big[\prod_{k<\ell}^{L} (1 - \mathbf{O}_k) \Big] \odot \mathbf{O}_\ell \odot \mathbf{C}_\ell. \tag{4}$$

This formulation has a clear interpretation where color maps are overlaid on top of each other with a transparency corresponding to their occupancy map. Note that we choose to drop the explicit depth dependency as all 3D coordinates (including depth) of a vertex already receive gradients by 3D-to-2D projection. Our layered aggregation used together with the SoftRasterizer's occupancy function $\mathcal{O}_{\text{SR}}$ results in face inner-borders that are visually unpleasant. We thus instead use the occupancy function introduced in [3] defined by:

$$\mathcal{O}(i, j) = \exp(\min(0, \frac{\nu(i, j)}{\sigma})). \tag{5}$$

Compared to $\mathcal{O}_{\text{SR}}$, this function yields constant occupancy of 1 inside the faces. In addition to its simplicity, our differential renderer has two main advantages compared to SoftRasterizer. First, gradients can directly flow through occupancies $\mathbf{O}_{1:L}$ and the vertex positions can be updated by photometric comparison. Second, our formulation involves only one hyperparameter (σ) instead of five, making it easier to use.

2 Model insights

2.1 Progressive conditioning

Figure 1 shows the results obtained on CompCars [18] at the end of each stage of the training. Given an input image (leftmost column), we show for each

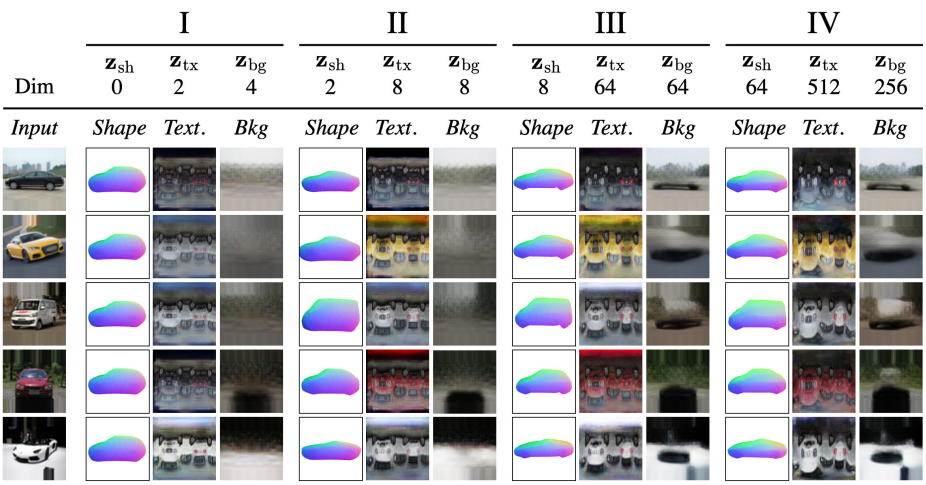


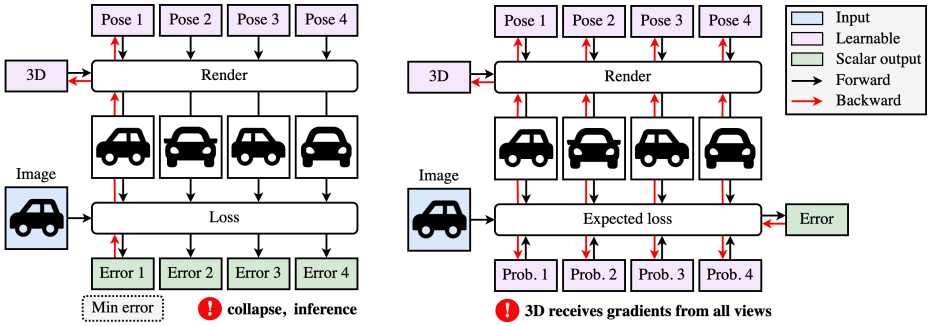
Fig. 1: **Progressive conditioning on CompCars** [18]**.** Given an input image (leftmost column), we show for each training stage, from left to right, a side view of the predicted shape, the texture image and the background image.

training stage the predicted outputs. From left to right, they correspond to a side view of the shape, the texture image and the background image. We can observe that all shape, texture and background models gradually specialize to the instance represented in the input. In particular, this allows us to start with a weak background model to avoid degenerate solutions and to end up with a powerful background model to improve the reconstruction quality. Also note how all the texture images are aligned.

2.2 Neighbor reconstruction

When computing the neighbor reconstructions, we explicitly find neighbors that have a viewpoint different from the predicted viewpoint. More specifically, for a given input, we compute the angle between the predicted rotation matrix and all rotation matrices of the memory bank. Following standard conventions, such an angle lies in $[0°, 180°]$. Then, we select a target angle range as follows: we split the range of angles $[20°, 180°]$ into a partition of V uniform and continuous bins, and we uniformly sample one of the V angle ranges. Finally, we look for neighbors in the subset of instances having an angle within the selected range. In all experiments, we use $V = 5$.

We use a total angle range of $[20°, 180°]$ instead of $[0°, 180°]$ to remove instances having a similar pose. Note that we first tried to find neighbors of different poses without further constraint (which amounts to using $V = 1$) but we found that learned latent codes were specialized by viewpoints, *e.g.*, front / back view images corresponding to a shape mode with unrealistic side views, and side view images corresponding to a shape mode with unrealistic front / back views.



(a) Optimization in [8] (b) Optimization in [4, 7, 17]

Fig. 2: **Prior optimizations for joint 3D/pose learning.**

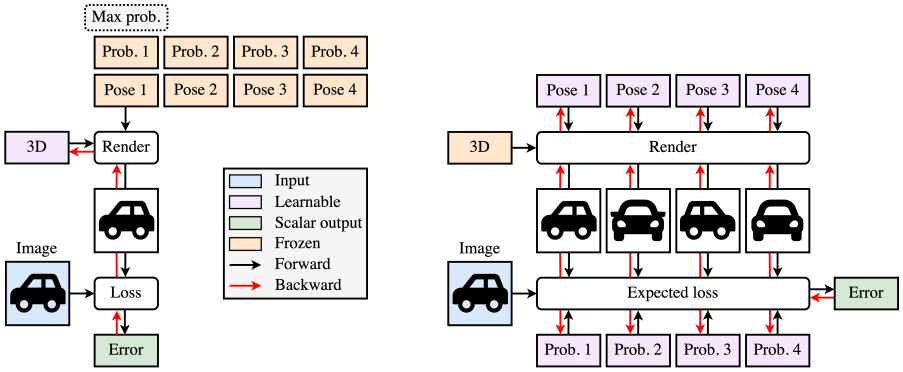
2.3 Joint 3D and pose learning

We analyze prior works on joint 3D and pose learning, illustrated in Figure 2, and compare them with our proposed optimization scheme, illustrated in Figure 3. Prior optimization schemes can be split in two groups: (i) learning through the minimal error reconstruction [8], and (ii) learning through an expected error [4, 7, 17].

In [8], all reconstructions associated to the different pose candidates are computed and both 3D and poses are updated using the reconstruction yielding the minimal error (see Figure 2a). We identified two major issues. First, because the other poses are not updated for a given input, we observed that a typical failure case corresponds to a collapse mode where only a single pose (or a small subset of poses) is used for all inputs. Indeed, there is no particular constraint that encourages the use of all pose candidates. Second, inference is not efficient as the object has to be rendered from all poses to find the correct object pose.

In [4, 7, 17], 3D and poses are updated using an expected reconstruction loss (see Figure 2b). While this allows to constrain the use of all pose candidates with a regularization on the predicted probabilities, we identified one major weakness common to these frameworks. Because the 3D receives gradients from all views, we observed a typical failure case where the 3D tries to fit the target input from all pose candidates yielding inaccurate texture and geometry. We argue such behaviour was not observed in previous works as they typically use a symmetry prior which prevents it from happening. Note that CMR [4] proposes to directly optimize for each training image a set of parameters corresponding to the pose candidates. This procedure not only involves memory issues as the number of parameters scales linearly with the number of training images, but also inference problems for new images. To mitigate the issue, they propose to use the learned poses as ground-truth to train an additional network that performs pose estimation given a new image.

In contrast, our proposed alternate optimization, illustrated in Figure 3, leverages the best of both worlds: (i) 3D receives gradients from the most likely



(a) 3D-step (b) P-step

Fig. 3: **Our alternate 3D / pose optimization.** Compared to prior works, we propose an optimization that alternates between 2 steps. **(a)** We update the 3D using the most likely pose candidate (3D-step). **(b)** We update the pose candidates and associated probabilities using the expected loss (P-step).

reconstruction, and (ii) all poses are updated using an expected loss. In practice, we alternate the optimization every new batch of inputs, and we define one iteration as either a a 3D-step or a P-step.

3 Quantitative evaluation

3.1 ICP alignment for better 3D evaluation

In the main paper, we align shapes using our gradient-based version of the Iterative Closest Point (ICP) [1] with anisotropic scaling before evaluating 3D reconstructions. For consistency, we use the same protocol across benchmarks and advocate to do so for future comparisons. First, meshes are centered and normalized so that they exactly fit inside the cube of unit length $[-0.5, 0.5]^3$; this is important to obtain results that are comparable. Second, we sample 100k points on the mesh surfaces. Third, we run our ICP implementation which minimizes by gradient descent the Chamfer-L_2 distance between the point clouds by jointly optimizing 3 translation parameters, 6 rotation parameters [19] and 3 scaling parameters. In practice, we use Adam optimizer [10], a learning rate of 0.01 and 100 iterations. Note that we use this gradient-based version of ICP instead of the classical iterative formulation as we found it to diverge when optimizing scale.

We argue that an ICP pre-processing is crucial for an unbiased 3D reconstruction evaluation and provide real examples in Figure 4 to support our claim. Rows correspond to different transformations of the same canonical shape, and for each row, we show: the transformation used, the resulting 3D shape, a rendering example as well as Chamfer-L_1 distance to the canonical shape. We overlay the visuals with green contours representing the canonical shape and the canonical rendering for easier comparisons. We can make two important observations. First,

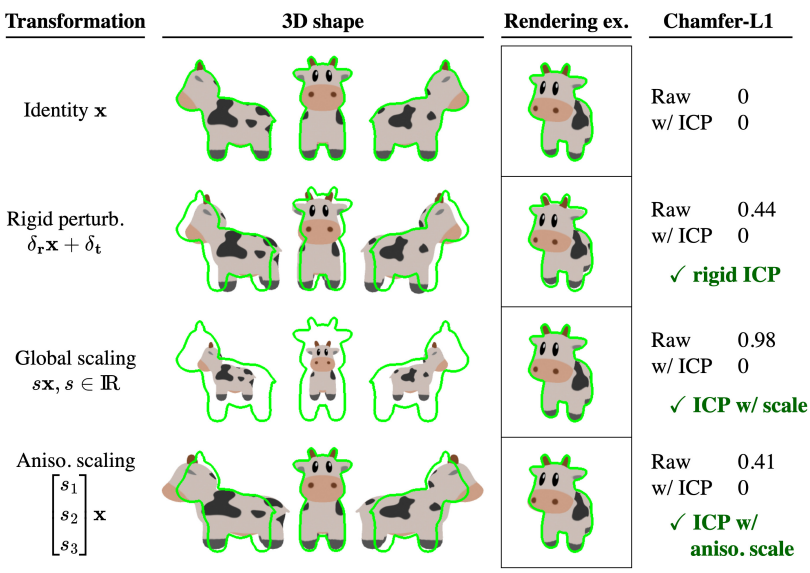


Fig. 4: **3D reconstruction evaluation with and without ICP.** Rows correspond to results obtained for transformed versions of a canonical shape and columns correspond to, from left to right, the transformation used, resulting 3D shape, a rendering example and Chamfer-L_1 distance to the canonical shape. Green contours represent the shape and rendering from the canonical object for visual comparisons.

although all the transformed shapes are excellent 3D reconstructions of the canonical shape, they result in dramatically poor performances. As a comparison, these performances are similar to our ShapeNet results with ICP when the model outputs degenerate reconstructions. Pre-processing the shapes using an ICP with anisotropic scaling mitigates this issue. Second, as shown by the rendering examples, for all these different shapes, we can find a pose that yields almost identical renderings. This hence emphasizes the numerous shape/pose ambiguities that arise from a given rendered image. As a result, it is extremely unlikely that a fully unsupervised SVR system predicting from a single image both the 3D shape and the pose will recover the exact pair of shape/pose used for annotations. In this case, the cameras used for rendering are the same and we do not even consider focal variations, which raises even more ambiguities.

3.2 ShapeNet results without ICP

For completeness, we provide quantitative results obtained without ICP on the ShapeNet benchmark in Table 1. We indicate the supervision used and visually separate methods using multi-view supervision. In addition to methods compared in the main paper, we report (i) results from category-agnostic versions (Cat. agn) of DVR [16] and SoftRas [11] presented in [16] and (ii) divergence results obtained by removing silhouette supervision from DVR and SoftRas.

Method	**Ours**	SDF-SRN	DVR	DVR	SoftRas	DVR	DVR	SoftRas
Cat. agn.				✓	✓		✓	✓
MV				✓	✓	✓	✓	✓
CK		✓	✓	✓	✓	✓	✓	✓
S		✓	✓			✓	✓	✓
airplane	0.186	0.173	**0.157**	Div.	Div.	0.151	0.190	**0.149**
bench	**0.257**	-	0.386	Div.	Div.	0.232	**0.210**	0.241
cabinet	**0.284**	-	0.849	Div.	Div.	0.257	**0.220**	0.231
car	0.251	**0.177**	0.282	Div.	Div.	0.198	**0.196**	0.221
chair	0.543	**0.333**	0.464	Div.	Div.	**0.249**	0.264	0.338
display	**0.344**	-	0.968	Div.	Div.	0.281	**0.255**	0.284
lamp	0.987	-	**0.688**	Div.	Div.	0.386	0.413	**0.381**
phone	**0.456**	-	1.412	Div.	Div.	0.147	0.148	**0.131**
rifle	**0.504**	-	0.528	Div.	Div.	**0.131**	0.175	0.155
sofa	**0.335**	-	0.665	Div.	Div.	**0.218**	0.224	0.407
speaker	**0.356**	-	0.535	Div.	Div.	0.321	**0.289**	0.320
table	**0.351**	-	0.442	Div.	Div.	0.283	**0.280**	0.374
vessel	**0.384**	-	0.400	Div.	Div.	**0.220**	0.245	0.233
mean	**0.403**	-	0.598	Div.	Div.	**0.236**	0.239	0.266

Table 1: **ShapeNet comparison without ICP.** We report Chamfer-L_1 ↓, supervisions are: Multi-Views, Camera or Keypoints, Silhouettes. We separate methods using multi-views and **best** results are highlighted in each group.

4 Implementation details

4.1 Modeling

Network architecture. We use the same neural network architecture for all experiments. The encoder is composed of 4 CNN backbones followed by separate Multi-Layer Perceptron (MLP) heads predicting a rendering parameter. More specifically, the 4 backbones are respectively used to predict: (i) shape code $\mathbf{z}_{\text{sh}}$ and scale $\mathbf{s}$; (ii) texture code $\mathbf{z}_{\text{tx}}$; (iii) background code $\mathbf{z}_{\text{bg}}$; (iv) rotations $\mathbf{r}_{1:K}$, translations $\mathbf{t}_{1:K}$, and pose probabilities $\mathbf{p}_{1:K}$. Note that using a shared backbone instead of separated ones also yields great results and is advocated for decreasing the memory footprint and training time; the major benefit from using separated backbones is to produce slightly more detailed textures and background. We follow prior works in SVR [4, 5, 12, 16] and use randomly initialized ResNet-18 [6] as backbone. Each MLP head has the same architecture with 3 hidden layers of 128 units and ReLU activations. The last layer of the MLP heads for shape, texture and background codes is initialized to zero to avoid discontinuity when increasing the size of the latent codes. The final activation of the MLP heads for scale, rotation, and translation is a tanh function and the output is scaled and shifted using predefined constants in order to control their range (see Table 2 for selected ranges). The learnable parts of the decoder are the shape deformation network s_θ and the two CNN generators t_θ and b_θ which respectively output 64×64 images for texture and background. The MLP modeling the deformations has 3 hidden layers, ReLU activations and 128 units for real images; we use an increased number of units for ShapeNet (512) which provides a small boost in performances. The CNN generators share the same architecture which is identical to the generator used in GIRAFFE [15]. We refer the reader to [15] for details.

Design type	ShapeNet	Real-image
ellipsoid scale	0.4	0.6
camera	$f = 3.732$	fov $= 30°$
$\mathbf{s}_x/\mathbf{s}_y/\mathbf{s}_z$	1 ± 0.5	1 ± 0.3
$\mathbf{t}_x/\mathbf{t}_y$	0 ± 0.5	0 ± 0.3
$\mathbf{t}_z$ (depth)	2.732	2.732 ± 0.3
$\mathbf{r}_a$ (azimuth)	$[0°, 360°]$	$[0°, 360°]$
$\mathbf{r}_e$ (elevation)	$30°$	$[-10°, 30°]$
$\mathbf{r}_r$ (roll)	$0°$	$[-30°, 30°]$

Table 2: **Design choices.** Following standard practices [11, 16] on ShapeNet [2], we keep the default rendering values used to generate the images for the focal length f, the distance to the camera $\mathbf{t}_z$ and the elevation $\mathbf{r}_e$. For real images, we keep the classical value of 2.732 for the distance to the camera $\mathbf{t}_z$ and use a field of view (fov) of $30°$. Note that we did not finetune these parameters, they were selected once through visual comparisons on a toy example.

Other design choices. In all experiments, the predefined anisotropic scaling used to deform the icosphere into an ellipsoid is $[1, 0.7, 0.7]$. In Table 2, we detail other design choices that are specific to all categories of ShapeNet [2] (second column) or all real-image datasets (third column). This notably includes a predetermined global scaling of the ellipsoid, a camera defined by a focal length f or a field of view (fov), as well as scaling, translation and rotation ranges.

4.2 Training

In all experiments, we use a batch size of 32 images of size 64×64 and Adam optimizer [10] with a constant learning rate of 10^{-4} that is divided by 5 at the very end of the training for a few epochs. The training corresponds to 4 stages where latent code dimensions are increased at the beginning of each stage and the network is then trained until convergence. We use dimensions 0/2/8/64 for the shape code, 2/8/64/512 for the texture code, and 4/8/64/256 for the background code if any. In line with the curriculum modeling of [13], we found it beneficial for the first stage to gradually increase the model complexity: we first learn to position the fixed ellipsoid in the image, then we allow the ellipsoid to be deformed, and finally we allow scale variabilities. In particular, we found this procedure prevents the model to learn prototypical shapes with unrealistic proportions. In the following, we describe other training details specific to ShapeNet [2] benchmark and real-image datasets.

ShapeNet dataset. We use the same training strategy for all categories. We train the first stage for 50k iterations, and each of the other stage for 250k iterations, where one iteration corresponds to either a 3D-step or a P-step of our alternate optimization. We do not learn a background model as all images are rendered on top of a white background. However, we found that our system learned in such synthetic setting was prone to a bad local minimum where

the predicted textures have white regions that accommodate for wrong shape prediction. Intuitively, this is expected as the system has no particular signal to distinguish white background regions from white object parts. To mitigate the issue, we constrain our texture model as follows: (i) during the first stage, the predicted texture image is averaged to yield a constant texture, and (ii) during the other stages, we occasionally use averaged textures instead of the real ones. More specifically, we sample a Bernoulli variable with probability $p = 0.2$ at each iteration and average the predicted texture image in case of success. We found this simple procedure to work well to resolve such shape/texture ambiguity.

Real-image datasets. We use the same training strategy for all real-image datasets. We train each stage for roughly 750k iterations, where one iteration either corresponds to a 3D-step or a P-step of our alternate optimization. Learning our structured autoencoder in such real-image scenario, without silhouette nor symmetry constraints, is very challenging. We found our system sometimes falls into a bad local minimum where the texture model is specialized by viewpoints, *e.g.*, dark cars always correspond to a frontal view and light cars always correspond to a back view. To alleviate the issue, we encourage uniform textures by occasionally using averaged textures instead of the real ones during rendering, as done on the ShapeNet benchmark. More specifically, we sample a Bernoulli variable with probability $p = 0.2$ at each iteration and average the predicted texture image in case of success. We observed that it was very effective in practice, and we also found it helped preventing the object texture from modeling background regions. We do not use such technique in the last stage to increase the texture accuracy.

References

1. Besl, P., McKay, N.D.: A method for registration of 3-D shapes. TPAMI **14**(2) (1992) 5
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012 [cs] (2015) 8
3. Chen, W., Gao, J., Ling, H., Smith, E.J., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In: NeurIPS (2019) 2
4. Goel, S., Kanazawa, A., Malik, J.: Shape and Viewpoint without Keypoints. In: ECCV (2020) 4, 7
5. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In: CVPR (2018) 7
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR (2016) 7
7. Henderson, P., Ferrari, V.: Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. IJCV (2019) 4
8. Insafutdinov, E., Dosovitskiy, A.: Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In: NIPS (2018) 4
9. Jojic, N., Frey, B.J.: Learning Flexible Sprites in Video Layers. In: CVPR (2001) 2
10. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: ICLR (2015) 5, 8
11. Liu, S., Li, T., Chen, W., Li, H.: Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In: ICCV (2019) 1, 6, 8
12. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy Networks: Learning 3D Reconstruction in Function Space. In: CVPR (2019) 7
13. Monnier, T., Groueix, T., Aubry, M.: Deep Transformation-Invariant Clustering. In: NeurIPS (2020) 8
14. Monnier, T., Vincent, E., Ponce, J., Aubry, M.: Unsupervised Layered Image Decomposition Into Object Prototypes. In: ICCV (2021) 2
15. Niemeyer, M., Geiger, A.: GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. In: CVPR (2021) 7
16. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In: CVPR (2020) 6, 7, 8
17. Tulsiani, S., Kulkarni, N., Gupta, A.: Implicit Mesh Reconstruction from Unannotated Image Collections. arXiv:2007.08504 [cs] (2020) 4
18. Yang, L., Luo, P., Loy, C.C., Tang, X.: A large-scale car dataset for fine-grained categorization and verification. In: CVPR (2015) 2, 3
19. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the Continuity of Rotation Representations in Neural Networks. In: CVPR (2020) 5