arXiv:2204.10310v1 [cs.CV] 21 Apr 2022

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

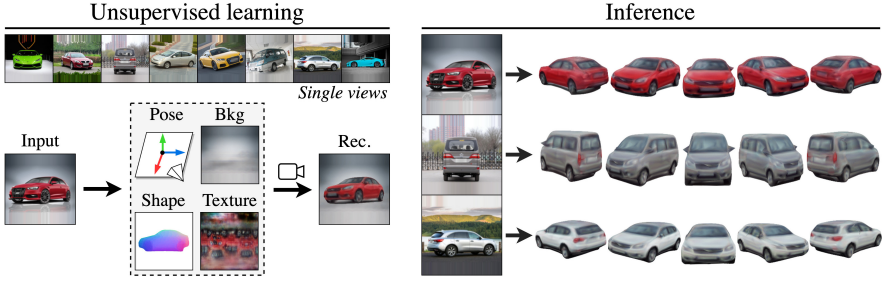


Fig. 1: **Single-View Reconstruction by Cross-Instance Consistency. (left)** Given a collection of single-view images from an object category, we learn without additional supervision an autoencoder that explicitly generates shape, texture, pose and background. **(right)** At inference time, our approach reconstructs high-quality textured meshes from raw single-view images.

Abstract. Approaches to single-view reconstruction typically rely on viewpoint annotations, silhouettes, the absence of background, multiple views of the same instance, a template shape, or symmetry. We avoid all of these supervisions and hypotheses by leveraging explicitly the consistency between images of different object instances. As a result, our method can learn from large collections of unlabelled images depicting the same object category. Our main contributions are two approaches to leverage cross-instance consistency: (i) *progressive conditioning*, a training strategy to gradually specialize the model from category to instances in a curriculum learning fashion; (ii) *swap reconstruction*, a loss enforcing consistency between instances having similar shape or texture. Critical to the success of our method are also: our structured autoencoding architecture decomposing an image into explicit shape, texture, pose, and background; an adapted formulation of differential rendering, and; a new optimization scheme alternating between 3D and pose learning. We compare our approach, UNICORN, both on the diverse synthetic ShapeNet dataset - the classical benchmark for methods requiring multiple views as supervision - and on standard real-image benchmarks (Pascal3D+ Car, CUB-200) for which most methods require known templates and silhouette annotations. We also showcase applicability to more challenging real-world collections (CompCars, LSUN), where silhouettes are not available and images are not cropped around the object.

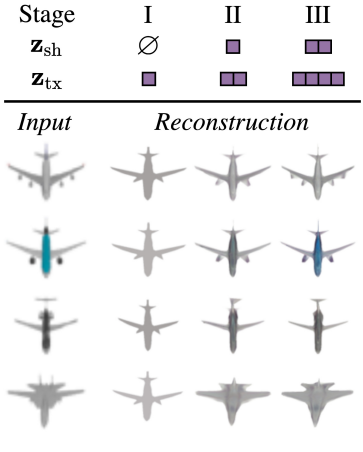
Keywords: unsupervised learning, single-view reconstruction

1 Introduction

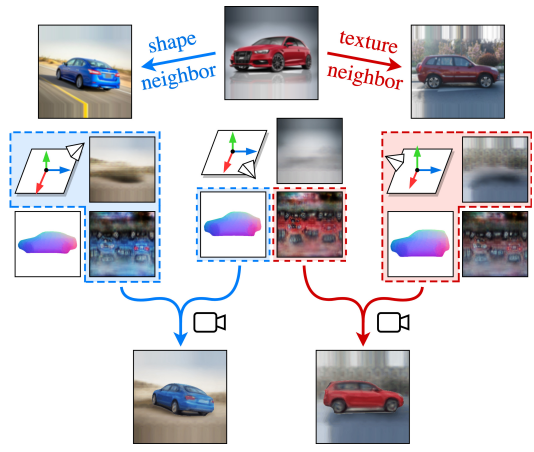
One of the most magical human perceptual abilities is being able to see the 3D world behind a 2D image – a mathematically impossible task! Indeed, the ancient Greeks were so incredulous at the possibility that humans could be "hallucinating" the third dimension, that they proposed the utterly implausible Emission Theory of Vision [9] (eye emitting light to "sense" the world) to explain it to themselves. In the history of computer vision, single-view reconstruction (inferring 3D from a single 2D image, or SVR) has had an almost cult status as one of the holy grail problems [17,18,38]. Recent advancements in data-driven deep learning methods have dramatically improved results in this area [6,12,30]. However, the best methods still require costly supervision, such as multi-view posed images [28,36]. Despite efforts to remove such requirements, the works with the least supervision still rely on two signals limiting their applicability: (i) silhouettes and (ii) strong priors such as symmetries [22], known template shapes [10,42], or the absence of background [49]. Although crucial to achieve reasonable results, priors like silhouette and symmetry limit applicability and reconstruction quality: silhouette annotations for real images are often coarse [5] and wrong symmetry predictions yield unrealistic reconstructions [49]. We propose a data-driven alternative to priors, silhouette annotations and multi-view supervision, which we demonstrate to be competitive for diverse datasets. Table 1 summarizes the comparison between our approach and selected prior works.

Method	Supervision	Synthetic data	Real data	Output
[6,12,30,45]★	**3D**	ShapeNet	✗	3D
[26,52]★	MV, C, S	ShapeNet	✗	3D
[5,28,36,43]★	MV, C, S	ShapeNet	✗	3D, T
[57]	MV, C, S	✗	Bird, Car, Horse	3D, T
[20,41]★	MV, S	ShapeNet	✗	3D, C
[23,43,44]★	CK, S	✗	Pascal	3D
[5,22]	CK, S, P(†)	✗	Bird, Car, Plane	3D, T
[16]	CK, P(†)	ShapeNet	Bird, Car	3D, T
[10]	S, P(♢, †)	✗	Bird, Car, Moto, Shoe	3D, T, C
[42]	S, P(♢, †)	✗	Animal, Car, Plane	3D, T, C
[27]	S, P(↔, †)	✗	Animal, Car, Moto	3D, T, C
[48]	S, P(‡)	✗	Vase	3D, T, C
[49]	P(☒, ≺, †)	✗	Face	D, T, C
[15]	P(☒, ∅)	Toy ShapeNet	✗	3D, C
Ours	None	ShapeNet	Animal, Car, Moto	3D, T, C

Table 1: **Comparison with selected works.** For each method, we outline the supervision and priors used (Multi-Views, Camera pose, Camera estimate or Keypoints, Silhouette, Priors like ♢ template shape, † symmetry, ‡ solid of revolution, ↔ semantic consistency, ☒ no/limited background, ≺ frontal view, ∅ no texture), which data it has been applied to and the corresponding outputs (3D, Depth, Texture, Camera pose). We mark category-agnostic models with ★.



(a) Progressive conditioning



(b) Swap reconstruction

Fig. 2: **Leveraging cross-instance consistency. (a)** Progressive conditioning amounts to gradually increasing, in a multi-stage fashion, the size of the conditioning latent spaces, here associated to shape $\mathbf{z}_{\text{sh}}$ and texture $\mathbf{z}_{\text{tx}}$. **(b)** We explicitly share the shape and texture models across neighboring instances by swapping their characteristics and applying a loss to associated swap reconstructions.

More precisely, we learn in an analysis-by-synthesis fashion a network that predicts for each input image a 3D shape parametrized as a deformation of an ellipsoid, a texture map, a camera viewpoint, and a background image. Our main insight to remove the hypotheses and supervisions required by other methods is to leverage consistency across different instances. First, we design a training procedure, *progressive conditioning*, which encourages the model to share elements between images by strongly constraining the variability of shape, texture and background at the beginning of training and progressively allowing for more diversity (Figure 2a). Second, we introduce a *swap reconstruction* loss, which explicitly enforces neighboring instances from different viewpoints to share the same shape or texture model (Figure 2b). Note that compared to works using symmetry priors to constrain the reconstruction of unseen parts [5, 10, 15, 22, 27, 42, 48, 49], these simple yet effective techniques are data driven and not specialized to any particular object or dataset.

We also provide two technical insights that we found critical to learn our model without viewpoint and silhouette annotations: (i) a new optimization strategy which alternates between learning a set of pose candidates with associated probabilities and learning all other components using the most likely candidate, and (ii) a differentiable rendering formulation inspired by layered image models [21, 32] which we found to perform better than the classical SoftRasterizer [28].

We validate our approach on the standard ShapeNet [4] benchmark, real-image SVR benchmarks (Pascal3D+ Car [50], CUB-200 [47]) as well as more complex real-world datasets (CompCars [53], LSUN Motorbike and Horse [54]). In all scenarios, we demonstrate high-quality and realistic textured 3D reconstructions.

Summary. We present UNICORN, an **UN**supervised framework using cross-**I**nstance **CO**nsistency for 3D **R**econstructio**N**. Our main contributions are:
– the first fully unsupervised SVR system, demonstrating state-of-the-art textured 3D reconstructions for both generic object shapes and real images, and not requiring hypothesis or supervision beyond a categorical image collection.
– two data-driven techniques to enforce cross-instance consistency: cross-instance consistency: *progressive conditioning* (Fig. 2a) and *swap reconstruction* (Fig. 2b).
Code and models are available at imagine.enpc.fr/~monniert/UNICORN.

2 Related work

In this section, we first review deep single-view reconstruction methods. We then present the mesh-based differential renderers we build upon. Finally, we discuss curriculum learning techniques to which our progressive conditioning is related.

Deep Single-View Reconstruction (SVR). There is a clear trend to remove supervision from deep SVR pipelines to directly learn 3D from raw 2D images, which we summarize in Table 1 and discuss below.

A first group of methods uses strong supervision, either paired 3D and images or multiple views of the same object. Direct 3D supervision is successfully used to learn voxels [6], meshes [45], parametrized surfaces [12] and implicit functions [30, 51]. The first methods using silhouettes and multiple views of the same instance initially require camera poses and are also developed for diverse 3D shape representations: [43, 52] opt for a voxel representation, [5, 26, 28] introduce mesh-based differentiable renderers, and [36] adapts implicit representations. Works like [20, 41] then introduce techniques to remove the assumption of known poses. Except for [57], which successfully leverages GAN-generated images [11, 24], these works are typically limited to synthetic datasets.

A second group of methods aims at removing the need for 3D or multi-view supervision. This is very challenging and they hence focus on learning 3D from images of a single category. Early works [23, 44] infer geometry by estimating camera poses with keypoints and minimizing the silhouette reprojection error. The ability to predict textures is first incorporated by CMR [22] which, in addition to keypoints and silhouettes, uses symmetry priors. [16] improves upon CMR and develops a framework for images with camera annotations that does not require silhouettes. Two works managed to further avoid the need for keypoints or camera estimates but at the cost of additional dataset-specific hypothesis: [15] shows results on synthetic images with textureless objects, [49] models 2.5D objects like faces with limited background and viewpoint variation. Finally, recent works only require object silhouettes but also make additional hypotheses: [10, 42] use known template shapes, [27] assumes access to an off-the-shelf system predicting part semantics, and [48] specifically targets solids of revolution.

We build on the insights from these works, but we *do not* use camera estimates, keypoints, silhouettes, nor strong dataset-specific priors, and demonstrate results for both diverse shapes and real-image collections. To the best of our knowledge, we present the first generic SVR system learned from raw image collections.

Mesh-based differentiable rendering. We represent 3D models as meshes with parametrized surfaces, as introduced in AtlasNet [12] and advocated by [42]. We optimize the mesh geometry, texture and camera parameters associated to an image using differentiable rendering. Loper and Black [29] introduce the first generic differentiable renderer by approximating derivatives with local filters, and [26] proposes an alternative approximation more suitable to learning neural networks. Another set of methods instead approximates the rendering function to allow differentiability, including SoftRasterizer [28,37] and DIB-R [5]. We refer the reader to [25] for a comprehensive study. We build upon SoftRasterizer [28], but modify the rendering function to better learn without silhouette information.

Curriculum learning. The idea of learning networks by "starting small" dates back to Elman [8] where two curriculum learning schemes are studied: (i) increasing the difficulty of samples, and (ii) increasing the complexity of the model. We respectively coin them *curriculum sampling* and *curriculum modeling* for differentiation. Known to drastically improve the convergence speed [2], curriculum sampling is widely adopted across various applications, such as representation learning [39], optical flow estimation [19] and natural language processing [1]. On the contrary, curriculum modeling is less studied although crucial to various methods. For example, [45] learns to perform SVR in a coarse-to-fine manner by increasing the number of mesh vertices, and [31] clusters images by aligning them with transformations that progressively increase in complexity. Notably, [31] quantitatively shows that such procedure dramatically improves results by avoiding bad local minima. In this work, we propose a new form of curriculum modeling dubbed *progressive conditioning* which also enables us to avoid bad local minima.

3 Approach

Our goal is to learn a neural network that reconstructs a textured 3D object from a single input image. We assume we have access to a raw collection of images depicting objects from the same category, without any further annotation. We propose to learn 3D in an analysis-by-synthesis fashion, by learning to autoencode images in a structured way as depicted in Figure 3. In this section, we first introduce our structured autoencoder (Section 3.1). We then present how we learn models consistent across instances (Section 3.2). Finally, we discuss one more technical contribution necessary to the learning of our model: an alternate optimization strategy for joint 3D and pose estimation (Section 3.3).

Notations. We use bold lowercase for vectors (e.g., $\mathbf{a}$), bold uppercase for images (e.g., $\mathbf{A}$), double-struck uppercase for meshes (e.g., $\mathbb{A}$), calligraphic uppercase for the main modules of our system (e.g., $\mathcal{A}$), lowercase indexed with generic parameters θ for networks (e.g., a_θ), and write $a_{1:N}$ the ordered set $\{a_1, \ldots, a_n\}$.

3.1 Structured autoencoding

Overview. Our approach can be seen as a structured autoencoder: it takes an image as input, computes parameters with an encoder, and decodes them into

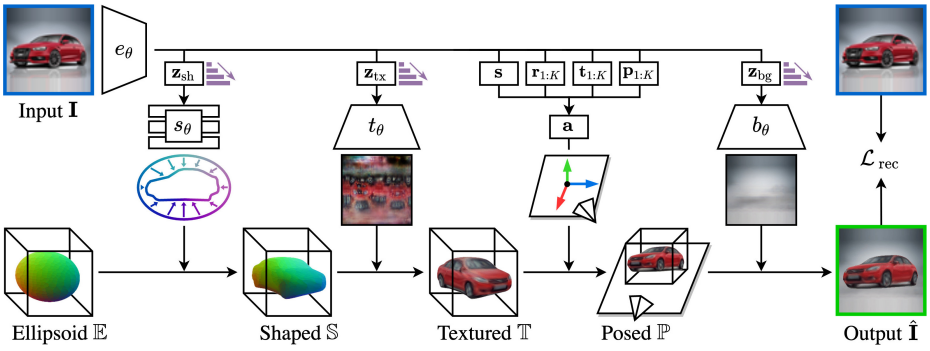


Fig. 3: **Structured autoencoding.** Given an input, we predict parameters that are decoded into 4 factors (shape, texture, pose, background) and composed to generate the output. Progressive conditioning is represented with .

explicit and interpretable factors that are composed to generate an image. We model images as the rendering of textured meshes on top of background images. For a given image $\mathbf{I}$, our model thus predicts a shape, a texture, a pose and a background which are composed to get the reconstruction $\hat{\mathbf{I}}$, as shown in Figure 3. More specifically, the image $\mathbf{I}$ is fed to a convolutional encoder network e_θ which outputs parameters $e_\theta(\mathbf{I}) = \{\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}, \mathbf{a}, \mathbf{z}_{\text{bg}}\}$ used for the decoding part. $\mathbf{a}$ is a 9D vector including the object pose, while the dimension of the latent codes $\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}$ and $\mathbf{z}_{\text{bg}}$ will vary during training (see Sec. 3.2). In the following, we describe the decoding modules using these parameters to build the final image by generating a shape, adding texture, positioning it and rendering it over a background.

Shape deformation. We follow [42] and use the parametrization of AtlasNet [12] where different shapes are represented as deformation fields applied to the unit sphere. We apply the deformation to an icosphere slightly stretched into an ellipsoid mesh $\mathbb{E}$ using a fixed anisotropic scaling. More specifically, given a 3D vertex $\mathbf{x}$ of the ellipsoid, our shape deformation module $\mathcal{S}_{\mathbf{z}_{\text{sh}}}$ is defined by $\mathcal{S}_{\mathbf{z}_{\text{sh}}}(\mathbf{x}) = \mathbf{x} + s_\theta(\mathbf{x}, \mathbf{z}_{\text{sh}})$, where $\mathbf{z}_{\text{sh}}$ is a shape code and s_θ is a Multi-Layer Perceptron (MLP) modeling a deformation field. Applying this displacement to all the ellipsoid vertices enables us to generate a shaped mesh $\mathbb{S} = \mathcal{S}_{\mathbf{z}_{\text{sh}}}(\mathbb{E})$. We found that using an ellipsoid instead of a raw icosphere was very effective in encouraging the learning of objects aligned w.r.t. the canonical axes. Learning surface deformations is often preferred to vertex-wise displacements as it enables mapping surfaces, and thus meshes, at any resolution. For us, the mesh resolution is kept fixed and such a representation is a way to regularize the deformations.

Texturing. Following the idea of CMR [22], we model textures as an image UV-mapped onto the mesh through the reference ellipsoid. More specifically, given a texture code $\mathbf{z}_{\text{tx}}$, a convolutional network t_θ is used to produce an image $t_\theta(\mathbf{z}_{\text{tx}})$, which is UV-mapped onto the sphere using spherical coordinates to associate a 2D point to every vertex of the ellipsoid, and thus to each vertex of the shaped mesh. We write $\mathcal{T}_{\mathbf{z}_{\text{tx}}}$ this module generating a textured mesh $\mathbb{T} = \mathcal{T}_{\mathbf{z}_{\text{tx}}}(\mathbb{S})$.

(a) Background overfits the input (b) 3D object overfits the input

Fig. 4: **Overfitting issues.** An SVR system learned by raw photometric autoencoding is prone to overfitting through **(a)** the background model or **(b)** the object model. We alleviate the issue with cross-instance consistency.

Affine transformation. To render the textured mesh $\mathbb{T}$, we define its position w.r.t. the camera. In addition, we found it beneficial to explicitly model an anisotropic scaling of the objects. Because predicting poses from raw photometry comparison is hard, we predict K 6D poses candidates, defined by rotations $\mathbf{r}_{1:K}$ and translations $\mathbf{t}_{1:K}$, and associated probabilities $\mathbf{p}_{1:K}$. This involves learning challenges we tackle with a specific optimization procedure described in Sec. 3.3. At inference, we select the pose with highest probability. We combine the scaling and the most likely 6D pose in an affine transformation module $\mathcal{A}_{\mathbf{a}}$. More precisely, $\mathcal{A}_{\mathbf{a}}$ is parametrized by $\mathbf{a} = \{\mathbf{s}, \mathbf{r}, \mathbf{t}\}$, where $\mathbf{s}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^3$ respectively correspond to the three scales of an anisotropic scaling, the three Euler angles of a rotation and the three coordinates of a translation. A 3D point $\mathbf{x}$ on the mesh is then transformed by $\mathcal{A}_{\mathbf{a}}(\mathbf{x}) = \text{rot}(\mathbf{r})\text{diag}(\mathbf{s})\mathbf{x} + \mathbf{t}$ where $\text{rot}(\mathbf{r})$ is the rotation matrix associated to $\mathbf{r}$ and $\text{diag}(\mathbf{s})$ is the diagonal matrix associated to $\mathbf{s}$. Our module is applied to all points of the textured mesh $\mathbb{T}$ resulting in a posed mesh $\mathbb{P} = \mathcal{A}_{\mathbf{a}}(\mathbb{T})$.

Rendering with background. The final step of our image formation process is to render the mesh over a background image. The background image is generated from a background code $\mathbf{z}_{\text{bg}}$ by a convolutional network b_θ. A differentiable module $\mathcal{B}_{\mathbf{z}_{\text{bg}}}$ renders the posed mesh $\mathbb{P}$ over this background image $b_\theta(\mathbf{z}_{\text{bg}})$ resulting in a reconstructed image $\hat{\mathbf{I}} = \mathcal{B}_{\mathbf{z}_{\text{bg}}}(\mathbb{P})$. We perform rendering through soft rasterization of the mesh. Because we observed that learning geometry from raw photometry with the standard SoftRasterizer [28,37] was hard, we propose two key changes: a layered aggregation of the projected faces and an alternative occupancy function. We provide details in our supplementary material.

3.2 Unsupervised learning with cross-instance consistency

We propose to learn our structured autoencoder without any supervision, by synthesizing 2D images and minimizing a reconstruction loss. Due to the unconstrained nature of the problem, such an approach typically overfits the input images (*e.g.*, Fig. 4a and Fig. 4b). While previous works leverage silhouettes and strong dataset-specific priors to mitigate this issue, we instead propose two unsupervised data-driven techniques, namely *progressive conditioning* (a training strategy) and *swap reconstruction* (a training loss). We thus optimize the shape, texture and background by minimizing for each image $\mathbf{I}$ reconstructed as $\hat{\mathbf{I}}$:

$$\mathcal{L}_{\text{3D}} = \mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_{\text{swap}}\mathcal{L}_{\text{swap}}(\mathbf{I}) + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}, \tag{1}$$

where λ_{swap} and λ_{reg} are scalar hyperparameters, and $\mathcal{L}_{\text{rec}}$, $\mathcal{L}_{\text{swap}}$ and $\mathcal{L}_{\text{reg}}$ are respectively the reconstruction, swap reconstruction, and regularization losses, described below. In all experiments, we use $\lambda_{\text{swap}} = 1$ and $\lambda_{\text{reg}} = 0.01$. Note that we optimize pose prediction using a slightly different loss in an alternate optimization scheme described in Section 3.3.

Reconstruction and regularization losses. Our reconstruction loss has two terms, a pixel-wise squared L_2 loss $\mathcal{L}_{\text{pix}}$ and a perceptual loss [56] $\mathcal{L}_{\text{perc}}$, an L_2 loss on the `relu3_3` layer of a pre-trained VGG16 network [40], similar to [49]. While pixel-wise losses are common for autoencoders, we found it crucial to add a perceptual loss to learn textures that are discriminative for our pose estimation. Note that we tried SSIM [46] as perceptual distance which also improved learning but yielded slightly worse results. Our full reconstruction loss can be written $\mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}) = \mathcal{L}_{\text{pix}}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}}(\mathbf{I}, \hat{\mathbf{I}})$ and we use $\lambda_{\text{perc}} = 10$ in all experiments. While our deformation-based surface parametrization naturally regularizes the shape, we observe it can fall into bad minima where the surface has folds and miss-oriented parts. Following prior works [5, 10, 28, 55], we thus add a small regularization term $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{lap}}$ consisting of a normal consistency loss [7] $\mathcal{L}_{\text{normal}}$ and a Laplacian smoothing loss [33] $\mathcal{L}_{\text{lap}}$.

Progressive conditioning. The goal of *progressive conditioning* is to encourage the model to share elements (*e.g.* shape, texture, background) across instances to prevent overfitting. Inspired by the curriculum learning philosophy [8, 31, 45], we propose to do so by gradually increasing the latent space representing the shape, texture and background. Intuitively, restricting the latent space implicitly encourages maximizing the information shared across instances. For example, a latent space of dimension 0 (*i.e.*, no conditioning) amounts to learning a global representation that is the same for all instances, while a latent space of dimension 1 restricts all the generated shapes, textures or backgrounds to lie on a 1-dimensional manifold. Progressively increasing the size of the latent code during training can be interpreted as gradually specializing from category-level to instance-level knowledge. Because common neural network implementations have fixed-size inputs, we implement progressive conditioning by masking out, stage-by-stage, a decreasing number of values of the latent code. Figure 2a illustrates the procedure with an example where we can observe the progressive specialization to particular instances: reactors gradually appear/disappear, textures get more accurate. All our experiments share the same 4-stage training strategy where the latent code dimension is increased at the beginning of each stage and the network is then trained until convergence. We use dimensions 0/2/8/32 for the shape code, 2/8/32/128 for the texture code and 4/16/64/128 for the background code. We provide result examples for each stage in our supplementary material.

Swap reconstruction. The idea behind *swap reconstruction* is to explicitly enforce consistency between different instances. Our key assumption is that neighboring instances with similar shape or texture exist in the dataset. If such neighbors are correctly identified, switching their shape or texture in our generation process should give similar reconstruction results. We hence propose

to swap characteristics from neighboring instances and apply our reconstruction loss on the associated reconstructions. Intuitively, this process can be seen as mimicking a multi-view supervision without actually having access to multi-view images by finding neighboring instances in well-designed latent spaces. Figure 2b illustrates the process with an example.

More specifically, let $\{\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}, \mathbf{a}, \mathbf{z}_{\text{bg}}\}$ be the parameters predicted by our encoder for a given input training image $\mathbf{I}$, let Ω be a memory bank storing the images and parameters of the last M instances processed by the network. We write $\Omega^{(m)} = \{\mathbf{I}^{(m)}, \mathbf{z}_{\text{sh}}^{(m)}, \mathbf{z}_{\text{tx}}^{(m)}, \mathbf{a}^{(m)}, \mathbf{z}_{\text{bg}}^{(m)}\}$ each of these M instances and associated parameters. We first select the closest instance from the memory bank Ω in the texture (respectively shape) code space using the L_2 distance, $m_t = \text{argmin}_m \|\mathbf{z}_{\text{tx}} - \mathbf{z}_{\text{tx}}^{(m)}\|_2$ (respectively $m_s = \text{argmin}_m \|\mathbf{z}_{\text{sh}} - \mathbf{z}_{\text{sh}}^{(m)}\|_2$). We then swap the codes and generate the reconstruction $\hat{\mathbf{I}}_{\text{tx}}^{(m_t)}$ (respectively $\hat{\mathbf{I}}_{\text{sh}}^{(m_s)}$) using the parameters $\{\mathbf{z}_{\text{sh}}^{(m_t)}, \mathbf{z}_{\text{tx}}, \mathbf{a}^{(m_t)}, \mathbf{z}_{\text{bg}}^{(m_t)}\}$ (respectively $\{\mathbf{z}_{\text{sh}}, \mathbf{z}_{\text{tx}}^{(m_s)}, \mathbf{a}^{(m_s)}, \mathbf{z}_{\text{bg}}^{(m_s)}\}$). Finally, we compute the reconstruction loss between the images $\mathbf{I}^{(m_t)}$ and $\hat{\mathbf{I}}_{\text{tx}}^{(m_t)}$ (respectively $\mathbf{I}^{(m_s)}$ and $\hat{\mathbf{I}}_{\text{sh}}^{(m_s)}$). Our full swap loss can thus be written:

$$\mathcal{L}_{\text{swap}}(\mathbf{I}) = \mathcal{L}_{\text{rec}}(\mathbf{I}^{(m_t)}, \hat{\mathbf{I}}_{\text{tx}}^{(m_t)}) + \mathcal{L}_{\text{rec}}(\mathbf{I}^{(m_s)}, \hat{\mathbf{I}}_{\text{sh}}^{(m_s)}). \tag{2}$$

Note that we recompute the parameters of the selected instances with the current network state, to avoid uncontrolled effects of changes in the network state.

To prevent latent codes from specializing by viewpoint, we additionally split the viewpoints into V bins w.r.t. the rotation angle, sample uniformly a target viewpoint bin for each input instance and look for the nearest instances only in the subset of instances within the target viewpoint bin. In all experiments, unless mentioned otherwise, we use $V = 4$ and a memory bank of size $M = 1024$.

3.3 Alternate 3D and pose learning

Because predicting 6D poses is hard due to self-occlusions, we follow prior works [10, 15, 20, 42] and predict multiple pose candidates and their likelihood. However, we identified failure modes in the standard optimization framework (detailed in our supplementary material) and instead propose a new optimization that alternates between 3D and pose learning. More specifically, given an input image $\mathbf{I}$, we predict K pose candidates $\{(\mathbf{r}_1, \mathbf{t}_1), \ldots, (\mathbf{r}_K, \mathbf{t}_K)\}$, and their associated probabilities $\mathbf{p}_{1:K}$. We render the model from the different poses, yielding K reconstructions $\hat{\mathbf{I}}_{1:K}$. We then alternate the learning between 2 steps: (i) the *3D-step* where shape, texture and background branches of the network are updated by minimizing $\mathcal{L}_{\text{3D}}$ using the pose associated to the highest probability, and (ii) the *P-step* where the branches of the network predicting candidate poses and their associated probabilities are updated by minimizing:

$$\mathcal{L}_{\text{P}} = \textstyle\sum_k \mathbf{p}_k \mathcal{L}_{\text{rec}}(\mathbf{I}, \hat{\mathbf{I}}_k) + \lambda_{\text{uni}} \mathcal{L}_{\text{uni}}, \tag{3}$$

where $\mathcal{L}_{\text{rec}}$ is the reconstruction loss described in Sec. 3.2, $\mathcal{L}_{\text{uni}}$ is a regularization loss on the predicted poses and λ_{uni} is a scalar hyperparameter. More precisely,

we use $\mathcal{L}_{\text{uni}} = \sum_k |\bar{\mathbf{p}}_k - 1/K|$ where $\bar{\mathbf{p}}_k$ is the averaged probabilities for candidate k in a particular training batch. Similar to [15], we found it crucial to introduce this regularization term to encourage the use of all pose candidates. In particular, this prevents a collapse mode where only one pose candidate is used. Note that we do not use the swap reconstruction loss which is not relevant for viewpoints. In all experiments, we use $\lambda_{\text{uni}} = 0.02$.

Inspired by the camera multiplex of [10], we parametrize rotations with the classical Euler angles (azimuth, elevation and roll) and rotation candidates correspond to offset angles w.r.t. reference viewpoints. Since in practice elevation has limited variations, our reference viewpoints are uniformly sampled along the azimuth dimension. Note that compared to [10], we do not directly optimize a set of pose candidates per training image, but instead learn a set of K predictors for the entire dataset. We use $K = 6$ in all experiments.

4 Experiments

We validate our approach in two standard setups. Our model is first quantitatively evaluated on the classical ShapeNet benchmark where competing methods use multiple views as supervision. Then, we compare it to state-of-the-art methods on standard real-image benchmarks and demonstrate its applicability to more complex datasets. Finally, we present an ablation study.

4.1 Quantitative evaluation on the ShapeNet benchmark

We compare our approach to state-of-the-art methods using multi-views, viewpoints and silhouettes as supervision. Our method is instead learned without supervision, on categorical image collections, *i.e.*, we train a model per class. For fair comparison, we train the best supervised method, DVR [36], in this setting.

We adhere to community standards [26,28,36] and use the rendering and splits from [26] of the ShapeNet dataset [4]. It corresponds to a subset of 13 classes of 3D objects, each object being rendered into a 64×64 image from 24 viewpoints uniformly spaced along the azimuth dimension. We evaluate results using the standard Chamfer-L_1 distance [30, 36]. Compared to competing methods having access to the ground-truth viewpoint during training, we predict it for each input image in addition to the 3D shape. This yields to shape/pose ambiguities and misalignment errors. We thus post-process the predicted shapes with the rigid Iterative Closest Point (ICP) [3] to align them w.r.t. the ground-truth shapes. For fair comparison, we evaluate the best supervised method, DVR [36], using this ICP post-processing.

We report our quantitative results and compare to the state-of-the-art methods in Table 2. We split the table in twogroups based on whether the method is category-specific (*i.e.* one model per class) or category-agnostic (*i.e.* one single model for all classes). We also indicate the use of the ICP alignment for evaluation. Our approach achieves results that are on average slightly worse but comparable to the state-of-the-art methods supervised with multiple views and viewpoint

Supervision	None	MV + C + S			
Method	**Ours**	DVR⋆	DVR⋆	DVR	SoftRas
Cat. specific	✓	✓	✓	-	-
ICP eval	✓	✓	-	-	-
airplane	**0.136**	0.139	0.151	0.190	**0.149**
bench	**0.208**	0.223	0.232	**0.210**	0.241
cabinet	**0.218**	0.235	0.257	**0.220**	0.231
car	0.213	**0.189**	0.198	**0.196**	0.221
chair	0.286	**0.224**	0.249	**0.264**	0.338
display	0.286	**0.216**	0.281	**0.255**	0.284
lamp	0.661	**0.346**	0.386	0.413	**0.381**
phone	0.152	**0.125**	0.147	0.148	**0.131**
rifle	0.119	**0.109**	0.131	0.175	**0.155**
sofa	0.248	**0.191**	0.218	**0.224**	0.407
speaker	0.353	**0.291**	0.321	**0.289**	0.320
table	0.301	**0.269**	0.283	**0.280**	0.374
vessel	0.252	**0.202**	0.220	0.245	**0.233**
mean	0.264	**0.212**	0.236	**0.239**	0.266

Table 2: **Comparisons on ShapeNet** [4]**.** Following standard practices [30, 36], we report the Chamfer-L_1 distance, **best** results are highlighted in each group. We mark methods we ran ourselves using official implementations (⋆).

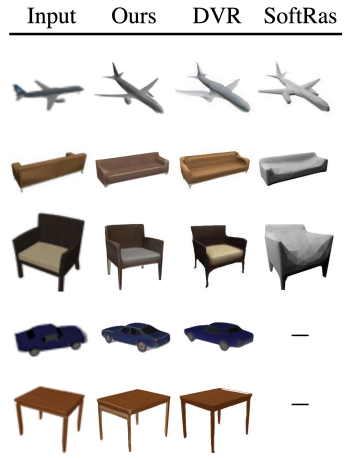


Fig. 5: **Qualitative comparisons.** We compare our results to DVR [36] and SoftRasterizer [28] for different ShapeNet [4] categories.

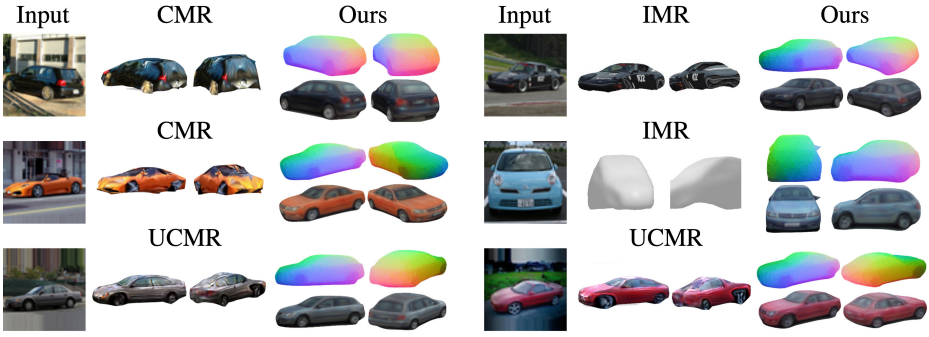
annotations. For some categories, it even yields slightly better performance. This is a strong result: it shows that our system learned on raw images generates 3D reconstructions comparable to the ones obtained from methods using geometry cues like silhouettes and multiple views. Note that for the lamp category, our method yields much worse results. We believe this is due mainly to the rotation invariance of many lamps, making our viewpoint estimation ambiguous.

We visualize and compare the quality of our 3D reconstructions in Figure 5. The first 3 examples correspond to examples advertised in DVR [36], the last 2 corresponds to examples we selected. Our method generates textured meshes of high-quality across all these categories. The geometry obtained is sharp and accurate, and the predicted texture mostly corresponds to the input image.

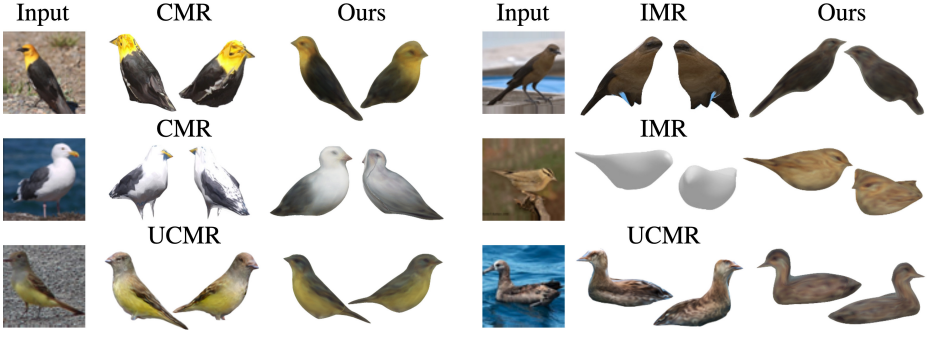
4.2 Qualitative results on real images

Pascal3D+ Car and CUB-200 benchmarks. We compare our approach to state-of-the-art SVR methods applied on real-image datasets, where multiple view annotations are not available. All competing methods use silhouette supervision and output meshes that are symmetric. CMR [22] additionally use keypoints, UCMR [10] and [42] starts learning from a given template shape of the category; we *do not* use any of these and directly learn from raw images.

We strictly follow the community standards [10, 22, 42]. We use the standard train/test splits of Pascal3D+ Car [50] and CUB-200 [47], respectively yielding 5000/220 images for Pascal3D+, and 6000/6000 images for CUB-200. Images are square-cropped around the object using the ground-truth bounding box and



(a) Pascal3D+ Car [50]



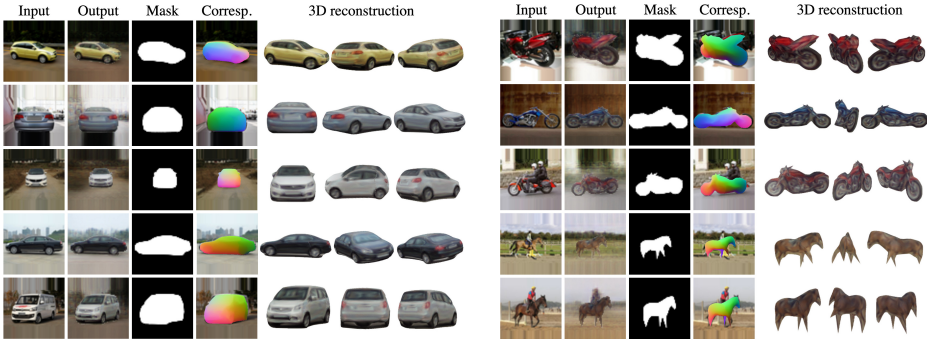
(b) CUB-200 [47]

Fig. 6: **Qualitative comparisons.** We show our reconstruction results for Pascal3D+ Car (top) and CUB-200 (bottom). For each benchmark, we compare to prior works: CMR [22] (top left), IMR [42] (top right), UCMR [10] (bottom).

resized to 64×64. Note that competing methods additionally use an off-the-shelf segmentation algorithm [13] to obtain foreground masks, which we do not use.

We qualitatively compare our approach to the state of the art for Pascal3D+ Car (Figure 6a) and CUB-200 (Figure 6b) benchmarks. All the examples shown are results advertised in the corresponding papers. For each input image, we show the textured mesh rendered from the predicted viewpoint (left) and a different viewpoint (right). For our car results, we additionally show the mesh with a synthetic texture to emphasize the correspondences found. Our approach yields results that outperform all prior works both in terms of geometric accuracy and overall realism. Although the textures obtained in [42] are more accurate, this is explained by the parametrization used where textures are modeled as a flow of the image pixels, which has a clear limitation, for example unseen texture parts are not realistic.

Real-word datasets. Motivated by 3D-aware image synthesis methods learned in-the-wild [34, 35], we investigate whether our approach can be applied to real-world datasets where silhouettes are not available and images are not methodically cropped around the object. We adhere to standards from the 3D-aware image



(a) CompCars [53] (b) LSUN Motorbike and Horse [54]

Fig. 7: **Real-world dataset results.** From left to right, we show for each input, the output image, the predicted mask, a correspondence map, and the mesh rendered from the predicted viewpoint and 2 other viewpoints. Note that for LSUN Horse, masks and correspondences are accurate but the geometry quality is low and outlines our approach limitations (see text). Best viewed digitally.

synthesis community [34, 35] and apply our approach to 64×64 images of CompCars [53]. In addition, we provide results for the more difficult scenario of LSUN images [54] for motorbikes and horses. Because many LSUN images are noise, we filter the datasets as follows: we manually select 16 reference images with different poses, find the nearest neighbors from the first 200k images in a pre-trained ResNet-18 [14] feature space, and keep the top 2k for each reference image. We repeat the procedure with flipped reference images yielding 25k images.

Our results are shown in Figure 7. For each input image, we show from left to right: the output image, the predicted mask, a correspondence map, and the 3D reconstruction rendered from the predicted viewpoint and 2 other viewpoints. Although our approach is trained to synthesize images, these are all natural by-products. While the quality of our 3D car reconstructions is high, the reconstructions obtained for LSUN images lack some realism and accuracy (especially for horses), thus outlining a limitation of our approach. We hypothesize this is mainly due to (i) the lack of front/back views in the data and (ii) the much more complex and diverse shapes the horse category involves. However, our segmentation and correspondence maps emphasize our system ability to accurately localize the object and find correspondences, without any supervision even when 3D reconstruction is not accurate.

4.3 Ablation study

We analyze the influence of progressive conditioning (PC) and swap reconstruction. We run experiments without each component and compare with the full model. In the following, we use $M = 256$.

First, we perform the ablation study on a subset of ShapeNet [4] and report results in Table 3. When $\mathcal{L}_{\text{swap}}$ is removed, the results are slightly worse for

Model	Full	w/o $\mathcal{L}_{\text{swap}}$	w/o PC
plane	**0.140**	0.159	0.143
bench	0.205	0.256	**0.204**
car	**0.222**	0.263	0.223
chair	**0.297**	0.466	0.626
mean	**0.216**	0.286	0.299

Table 3: **Ablation on ShapeNet** [4]**.**

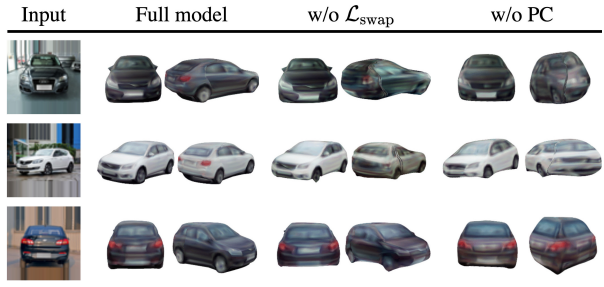


Fig. 8: **Ablation on CompCars** [53]**.** For each input, we show the mesh rendered from two viewpoints.

all categories, showing the swap reconstruction is important to the accuracy of the predicted geometry. When PC is removed, results are on par with the full model for planes, benches and cars, but much worse for chairs. Chairs have more complex shapes than other categories and our system falls into a bad local minimum where the object model overfits the input. Note that results in Tab. 2 correspond to a larger memory bank and are slightly better, thus emphasizing the influence of the memory bank size.

Second, we compare our method and its ablations on real images from CompCars [53] and visualize reconstruction examples in Figure 8. For each input image, we show the textured mesh rendered from the predicted viewpoint (left) and a different viewpoint (right). When $\mathcal{L}_{\text{swap}}$ is removed, we observe that the reconstruction quality from the predicted viewpoint remains high, but the one from another viewpoint decreases dramatically. Indeed, the swap reconstruction explicitly enforced the unseen reconstructed parts to be consistent with other instances. When PC is removed, we observe a clear case of overfitting where the reconstruction rendered from a different viewpoint does not correspond to a car.

5 Conclusion

We presented UNICORN, an unsupervised SVR method which, in contrast to all prior works, learns from raw images only. We demonstrated it yields high-quality results for diverse shapes as well as challenging real-world image collections. This was enabled by two key contributions aiming at leveraging consistency across different instances: our *swap reconstruction* loss and *progressive conditioning* training strategy. We believe our work includes both an important step forward for unsupervised SVR and the introduction of a valuable conceptual insight.

Acknowledgements

We thank François Darmon for inspiring discussions; Robin Champenois, Romain Loiseau, Elliot Vincent for feedback on the manuscript; and Michael Niemeyer, Shubham Goel for details on the evaluation. This work was supported in part by ANR project EnHerit ANR-17-CE23-0008, project Rapid Tabasco, gifts from Adobe and HPC resources from GENCI-IDRIS (2021-AD011011697R1).

References

1. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In: NIPS (2015) 5
2. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: ICML (2009) 5
3. Besl, P., McKay, N.D.: A method for registration of 3-D shapes. TPAMI **14**(2) (1992) 10
4. 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) 3, 10, 11, 13, 14, 18, 19
5. 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, 3, 4, 5, 8, 24
6. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In: ECCV (2016) 2, 4
7. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: SIGGRAPH (1999) 8
8. Elman, J.L.: Learning and development in neural networks: The importance of starting small. Cognition (1993) 5, 8
9. Finger, S.: Origins of neuroscience: a history of explorations into brain function. Oxford University Press (1994) 2
10. Goel, S., Kanazawa, A., Malik, J.: Shape and Viewpoint without Keypoints. In: ECCV (2020) 2, 3, 4, 8, 9, 10, 11, 12, 18, 21, 22
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Nets. In: NIPS (2014) 4
12. 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) 2, 4, 5, 6, 18
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017) 12
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR (2016) 13, 18
15. Henderson, P., Ferrari, V.: Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. IJCV (2019) 2, 3, 4, 9, 10, 21, 22
16. Henderson, P., Tsiminaki, V., Lampert, C.H.: Leveraging 2D Data to Learn Textured 3D Mesh Generation. In: CVPR (2020) 2, 4
17. Hoiem, D., Efros, A.A., Hebert, M.: Geometric Context from a Single Image. In: ICCV (2005) 2
18. Hoiem, D., Efros, A.A., Hebert, M.: Putting Objects in Perspective. IJCV (2008) 2
19. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In: CVPR (2017) 5
20. Insafutdinov, E., Dosovitskiy, A.: Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In: NIPS (2018) 2, 4, 9, 21
21. Jojic, N., Frey, B.J.: Learning Flexible Sprites in Video Layers. In: CVPR (2001) 3, 24
22. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning Category-Specific Mesh Reconstruction from Image Collections. In: ECCV (2018) 2, 3, 4, 6, 11, 12
23. Kar, A., Tulsiani, S., Carreira, J., Malik, J.: Category-Specific Object Reconstruction from a Single Image. In: CVPR (2015) 2, 4

24. Karras, T., Laine, S., Aila, T.: A Style-Based Generator Architecture for Generative Adversarial Networks. In: CVPR (2019) 4
25. Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., Gaidon, A.: Differentiable Rendering: A Survey. arXiv:2006.12057 [cs] (2020) 5
26. Kato, H., Ushiku, Y., Harada, T.: Neural 3D Mesh Renderer. In: CVPR (2018) 2, 4, 5, 10
27. Li, X., Liu, S., Kim, K., De Mello, S., Jampani, V., Yang, M.H., Kautz, J.: Self-supervised Single-view 3D Reconstruction via Semantic Consistency. In: ECCV (2020) 2, 3, 4
28. Liu, S., Li, T., Chen, W., Li, H.: Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In: ICCV (2019) 2, 3, 4, 5, 7, 8, 10, 11, 19, 23
29. Loper, M.M., Black, M.J.: OpenDR: An Approximate Differentiable Renderer. In: ECCV 2014, vol. 8695 (2014) 5
30. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy Networks: Learning 3D Reconstruction in Function Space. In: CVPR (2019) 2, 4, 10, 11, 18
31. Monnier, T., Groueix, T., Aubry, M.: Deep Transformation-Invariant Clustering. In: NeurIPS (2020) 5, 8, 19
32. Monnier, T., Vincent, E., Ponce, J., Aubry, M.: Unsupervised Layered Image Decomposition Into Object Prototypes. In: ICCV (2021) 3, 24
33. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Laplacian mesh optimization. In: GRAPHITE (2006) 8
34. Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: HoloGAN: Unsupervised learning of 3D representations from natural images. In: ICCV (2019) 12, 13
35. Niemeyer, M., Geiger, A.: GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. In: CVPR (2021) 12, 13, 18
36. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In: CVPR (2020) 2, 4, 10, 11, 18, 19
37. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3D Deep Learning with PyTorch3D. arXiv:2007.08501 [cs] (2020) 5, 7
38. Saxena, A., Min Sun, Ng, A.: Make3D: Learning 3D Scene Structure from a Single Still Image. TPAMI (2009) 2
39. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: CVPR (2015) 5
40. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: ICLR (2015) 8
41. Tulsiani, S., Efros, A.A., Malik, J.: Multi-view Consistency as Supervisory Signal for Learning Shape and Pose Prediction. In: CVPR (2018) 2, 4
42. Tulsiani, S., Kulkarni, N., Gupta, A.: Implicit Mesh Reconstruction from Unannotated Image Collections. arXiv:2007.08504 [cs] (2020) 2, 3, 4, 5, 6, 9, 11, 12, 21, 22
43. Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. In: CVPR (2017) 2, 4
44. Vicente, S., Carreira, J., Agapito, L., Batista, J.: Reconstructing PASCAL VOC. In: CVPR (2014) 2, 4
45. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In: ECCV (2018) 2, 4, 5, 8

46. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing (2004) 8
47. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-UCSD Birds 200. Tech. rep., California Institute of Technology (2010) 3, 11, 12
48. Wu, S., Makadia, A., Wu, J., Snavely, N., Tucker, R., Kanazawa, A.: De-rendering the World's Revolutionary Artefacts. In: CVPR (2021) 2, 3, 4
49. Wu, S., Rupprecht, C., Vedaldi, A.: Unsupervised Learning of Probably Symmetric Deformable 3D Objects from Images in the Wild. In: CVPR (2020) 2, 3, 4, 8
50. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond PASCAL: A benchmark for 3D object detection in the wild. In: WACV (2014) 3, 11, 12
51. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction. In: NeurIPS (2019) 4
52. Yan, X., Yang, J., Yumer, E., Guo, Y., Lee, H.: Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision. In: NeurIPS (2016) 2, 4
53. Yang, L., Luo, P., Loy, C.C., Tang, X.: A large-scale car dataset for fine-grained categorization and verification. In: CVPR (2015) 3, 13, 14, 20
54. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. arXiv:1506.03365 [cs] (2016) 3, 13
55. Zhang, J.Y., Yang, G., Tulsiani, S., Ramanan, D.: NeRS: Neural Reflectance Surfaces for Sparse-view 3D Reconstruction in the Wild. In: NeurIPS (2021) 8
56. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: CVPR (2018) 8
57. Zhang, Y., Chen, W., Ling, H., Gao, J., Zhang, Y., Torralba, A., Fidler, S.: Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering. In: ICLR (2021) 2, 4

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

In this supplementary document, we first provide implementation details (Appendix A), including network architectures, design choices and training details. Then, we present additional model insights related to progressive conditioning, swap reconstruction and 3D/pose optimization (Appendix B). Finally, we describe our custom differentiable rendering function (Appendix C).

A Implementation details

A.1 Modeling

Network architecture. We use the same neural network architecture for all experiments. The encoder is composed of a CNN backbone followed by separate Multi-Layer Perceptron (MLP) heads, each head predicting a rendering parameter namely shape code $\mathbf{z}_{\text{sh}}$, texture code $\mathbf{z}_{\text{tx}}$, background code $\mathbf{z}_{\text{bg}}$, scale $\mathbf{s}$, rotations $\mathbf{r}_{1:K}$,translations $\mathbf{t}_{1:K}$, and pose probabilities $\mathbf{p}_{1:K}$. More specifically, we follow prior works in SVR [10, 12, 30, 36] and use a ResNet-18 [14] as backbone. Each MLP has the same architecture with three 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 to predefined constants in order to control their range (see Table 4 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 the same architecture as the other MLPs (three hidden layers, 128 units, ReLU activations). The CNN generators share the same architecture which is identical to the generator used in GIRAFFE [35]. We refer the reader to [35] for details.

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 4, we detail other design choices that are specific to all categories of ShapeNet [4] (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.

A.2 Training

In all experiments, we use a batch size of 32 images of size 64×64 and Adam optimizer with a constant learning rate of 10^{-4}. The training corresponds to

Design type	ShapeNet	Real-image
ellipsoid scale	0.4	0.6
camera	$f = 3.732$	fov $= 30^\circ$
$\mathbf{s}_x/\mathbf{s}_y/\mathbf{s}_z$	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)	$[0^\circ, 360^\circ]$	$[0^\circ, 360^\circ]$
$\mathbf{r}_e$ (elevation)	30°	$[-10^\circ, 30^\circ]$
$\mathbf{r}_r$ (roll)	0°	$[-30^\circ, 30^\circ]$

Table 4: **Design choices.** Following standard practices [28, 36] on ShapeNet [4], 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.

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 0/2/8/32 for the shape code, 2/8/32/128 for the texture code, and 4/16/64/128 for the background code if any. In line with the curriculum modeling of [31], 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 [4] 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 minima 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 500k 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 minima 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.

B Model insights

B.1 Progressive conditioning

Figure 9 shows the results obtained on CompCars [53] at the end of each stage of the training. Given an input image (leftmost column), we show for each 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 overfitting and to end up with a powerful background model to improve the reconstruction quality. Also note how all the texture images are aligned.

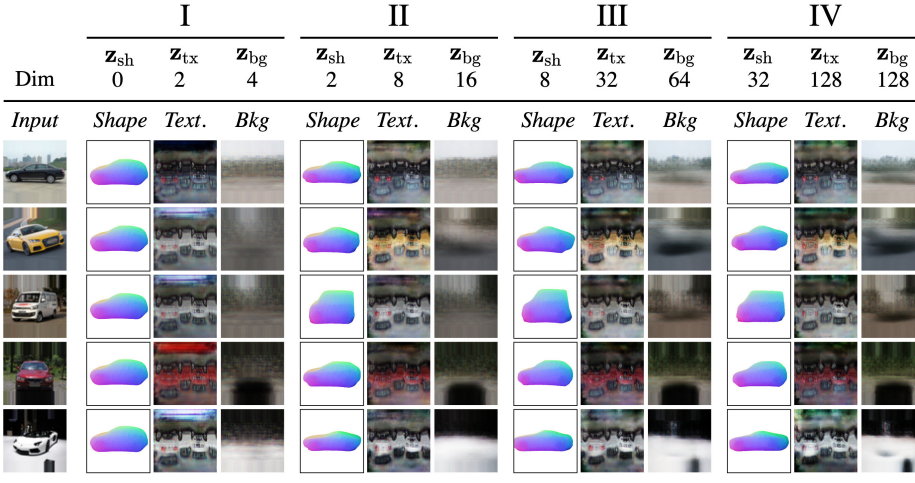


Fig. 9: **Progressive conditioning on CompCars [53].** Given an input image (leftmost column), we show for each training stage, from left to right, a side view of the predicted shape, the texture image and the background image.

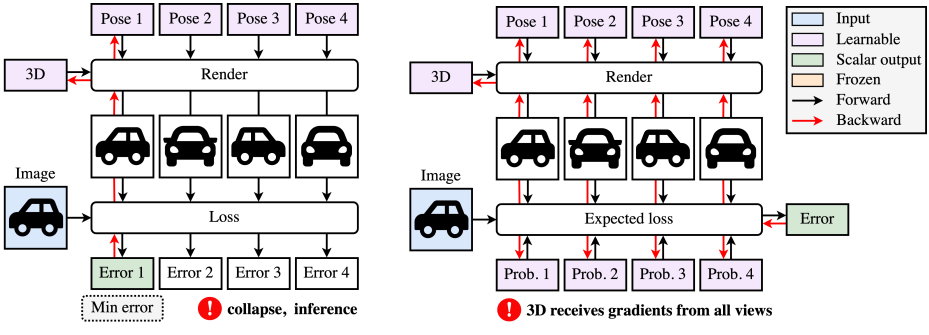
B.2 Swap reconstruction

When computing the swap 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 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 $V = 4$.

We use a total angle range of $[20°, 180°]$ instead of $[0°, 180°]$ to remove instances that have 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 are 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.

B.3 Joint 3D and pose learning

We analyze prior works on joint 3D and pose learning, illustrated in Figure 10, and compare them with our proposed optimization scheme, illustrated in Figure 11. Prior optimization schemes can be split in two groups: (i) learning through the minimal error reconstruction [20], and (ii) learning through an expected error [10, 15, 42].



(a) Optimization in [20] (b) Optimization in [10, 15, 42]

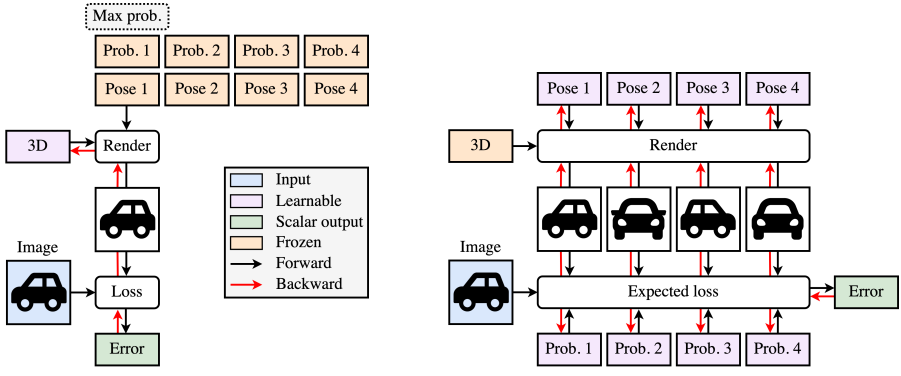
Fig. 10: **Prior optimizations for joint 3D/pose learning.**

In [20], 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 10a). 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 [10,15,42], 3D and poses are updated using an expected reconstruction loss (see Figure 10b). 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 [10] 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 11, leverages the best of both worlds: (i) 3D receives gradients from the most likely 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.



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

Fig. 11: **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).

C Differentiable rendering

Our output images correspond to the soft rasterization of a textured mesh on top of a background image. We observed that learning geometry from raw photometry with the standard SoftRasterizer [28] was hard and 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, 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:

$$\mathcal{O}_{\text{SR}}(i,j) = \text{sigmoid}\left(\frac{\nu(i,j)}{\sigma}\right), \tag{4}$$

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 an 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 $\hat{\mathbf{I}}$:

$$\mathcal{C}_{\text{SR}}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}, \mathbf{D}_{1:L}) = \sum_{\ell=1}^{L} \frac{\mathbf{O}_\ell \odot \exp(\mathbf{D}'_\ell/\gamma)}{\sum_k \mathbf{O}_k \odot \exp(\mathbf{D}'_k/\gamma)} \odot \mathbf{C}_\ell, \tag{5}$$

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 (5) 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 reconstruction. The simple case of a single face on a black background gives:

$$\hat{\mathbf{I}} = \frac{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}}{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma} + e^{\epsilon/\gamma}} \odot \mathbf{C}_1 \approx \frac{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}}{\mathbf{O}_1 \odot e^{\mathbf{D}'_1/\gamma}} \odot \mathbf{C}_1 = \mathbf{C}_1, \tag{6}$$

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 extreme 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 [21,32], 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:

$$\mathcal{C}(\mathbf{O}_{1:L}, \mathbf{C}_{1:L}) = \sum_{\ell=1}^{L} \Big[\prod_{k<\ell}^{L} (1 - \mathbf{O}_k) \Big] \odot \mathbf{O}_\ell \odot \mathbf{C}_\ell. \tag{7}$$

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 [5] defined by:

$$\mathcal{O}(i, j) = \exp(\min(0, \frac{\nu(i, j)}{\sigma})). \tag{8}$$

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. Second, our formulation involves only one hyperparameter (σ) instead of five, making it easier to use.