

Cycle-Consistent Generative Rendering for 2D-3D Modality Translation

Tristan Aumentado-Armstrong

tristan.a@partner.samsung.com

Alex Levenshtein

alex.lev@samsung.com

Stavros Tsogkas

stavros.t@samsung.com

Konstantinos G. Derpanis

k.derpanis@samsung.com

Allan D. Jepson

allan.jepson@samsung.com

Samsung AI Centre Toronto

Abstract

For humans, visual understanding is inherently generative: given a 3D shape, we can postulate how it would look in the world; given a 2D image, we can infer the 3D structure that likely gave rise to it. We can thus translate between the 2D visual and 3D structural modalities of a given object. In the context of computer vision, this corresponds to a learnable module that serves two purposes: (i) generate a realistic rendering of a 3D object (shape-to-image translation) and (ii) infer a realistic 3D shape from an image (image-to-shape translation). In this paper, we learn such a module while being conscious of the difficulties in obtaining large paired 2D-3D datasets. By leveraging generative domain translation methods, we are able to define a learning algorithm that requires only weak supervision, with unpaired data. The resulting model is not only able to perform 3D shape, pose, and texture inference from 2D images, but can also generate novel textured 3D shapes and renders, similar to a graphics pipeline. More specifically, our method (i) infers an explicit 3D mesh representation, (ii) utilizes example shapes to regularize inference, (iii) requires only an image mask (no keypoints or camera extrinsics), and (iv) has generative capabilities. While prior work explores subsets of these properties, their combination is novel. We demonstrate the utility of our learned representation, as well as its performance on image generation and unpaired 3D shape inference tasks.

1. Introduction

A natural approach to visual inference is the “analysis by synthesis” paradigm [81], which postulates that cognitive processing proceeds in a generative fashion [79]. This viewpoint can be referred to as the “inverse graphics” characterization of computer vision [41, 78, 75]. We expect algorithms of this type to extract a “graphics code” from a given image, which not only explains the input by reconstructing

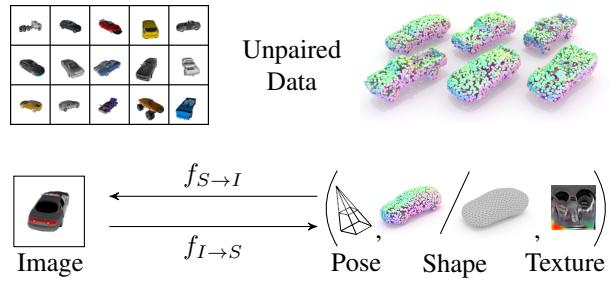


Figure 1. Depiction of our method for learning the 2D-3D *modality translation* functions, $f_{S \rightarrow I}$ and $f_{I \rightarrow S}$. Given unpaired collections of images and untextured 3D shapes (upper inset), we learn to generate images by rendering shapes with plausible textures and poses ($f_{S \rightarrow I}$), and infer 3D shape, pose, and texture from an image ($f_{I \rightarrow S}$), as shown in the lower inset. Our method takes as input an oriented point cloud in the shape-to-image direction, and infers a mesh representation in the image-to-shape direction.

it, but also constitutes an interpretable and manipulable latent representation of the image. Since the underlying world is three dimensional, such a representation should include 3D information as well. However, learning such a model is non-trivial. Even within a single category, the appearance, non-rigid shape, and pose of objects can vary widely. Furthermore, self-occlusions in the image make inferring the correct shape difficult, without a strong prior on the allowable shapes, even in relatively simple contexts.

From a modern learning perspective, the paucity of paired 2D-3D data presents an additional challenge to learning such inverse graphics models. While image categories can be annotated by humans with relative ease, efficiently obtaining and aligning the correct 3D model for an object into an image is difficult, even without considering issues like textures and lighting. On the other hand, synthetically generated paired datasets suffer from a domain shift, necessitating the use of techniques like domain randomization (DR), for helping models transfer from simulated to real settings [71, 70]. Though useful in many scenarios, DR images are still

not realistic, and the resulting domain mismatch may thus result in inappropriate data for some tasks. Separately, it is possible to construct data via standard generative models on 2D images (e.g., adversarial approaches [15]), but these do not have a naturally interpretable 3D latent representation for labelling or other downstream tasks.

In this paper, we present a weakly supervised model for learning from unpaired data, which only requires a set of images and untextured 3D shapes *without correspondence*. This allows us to leverage existing 3D datasets, despite a lack of explicit annotations relating shapes to images. Similar to cycle-consistent adversarial models, capable of domain translation between image types [82], we present an algorithm for translating between two representational *modalities* of objects: 3D shapes and 2D images of those shapes (see Fig. 1). In other words, our model encapsulates both the graphics rendering pipeline, mapping a 3D shape to an image, and the computer vision inference pipeline, mapping an image to its latent graphics code, *within a single learning framework*. The resulting conditional generative model is able to generate novel images, by generating a texture for an input shape and rendering the result; conversely, we can perform single-image 3D reconstruction, by explicitly inferring the 3D shape, pose, and appearance of an object from a given image. We use surface meshes as our 3D representation, as they circumvent problems with memory, resolution, manipulation, and rendering.

We enumerate our contributions as follows: (a) we define a cyclic generative model on unpaired 2D and 3D data, without camera calibration meta-information; (b) we show how to utilize adversarial methods to impose a strong prior on generated and inferred shapes, textures, poses, and renders; and (c) we demonstrate the effectiveness of our model on image generation, single-image textured shape inference, and representation learning, showing that our method can jointly understand the 2D and 3D modalities of an object.

2. Related Work

Inferring the 3D structure of an object from a single image is among the most studied problems in computer vision, and continues to receive considerable attention [18]. Several recent works have shown impressive results reconstructing meshes from 2D images using 3D supervision [76, 14, 57]. A different approach is to use powerful parametric models of deformations [48, 85], which makes it possible to do highly effective mesh reconstruction from 2D data [33, 84], but requires a pre-existing model of a canonical shape and its deformation space. Another recent trend is the weakening of supervision requirements for 3D inference, such as via an adversarial prior on the output shapes [32, 17, 23, 33]. One can also replace 3D supervision with multi-view information [74] or keypoints [34]. Tulsiani et al. [72] tackle the additional difficult problem of inferring pose from images,

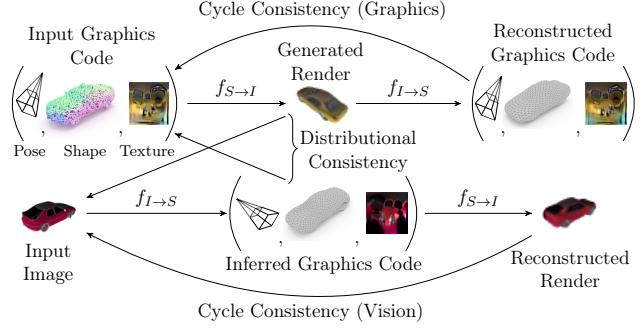


Figure 2. Overview of the cycle-consistent modality translation model. Our two modalities consist of (i) 2D images (assumed to be renders of 3D objects) and (ii) 3D representations, which we show as a *graphics code*, including pose, shape, and texture. For shorthand, we refer to this graphics code as simply “shape”. The top pathway depicts the *graphics cycle* (shape-to-image-to-shape), whereas the bottom one shows the *vision cycle* (image-to-shape-to-image), via our modality translation functions $f_{I \rightarrow S}$ and $f_{S \rightarrow I}$. The primary loss functions are also depicted: (a) the distributional consistency between (i) inferred versus input shapes, and (ii) generated versus real images; (b) the cyclic consistency between both the reconstructed and original input shapes and images. Note that the graphics cycle starts with a 3D point set, as well as random pose and texture samples, but the inferred and reconstructed codes utilize a mesh (see text for details).

as well as shape, utilizing only multi-view consistency for weak supervision. We infer both shape and pose from an image, using only unpaired shape data to form an adversarial prior on the inference process.

Relatively few machine learning models exist for generative models of textured 3D meshes [6, 22]. However, with the advent of differentiable renderers [36, 44, 46, 6], this has become more viable. Visual Object Networks [83] jointly model 3D shape and images in a generative fashion, but do not handle full 3D textures. Raj et al. [64] generate 3D textures by first texturing multiple 2.5D views, and combining the results in a differentiable renderer. Learned texture fields [56] are able to learn a 3D representation-agnostic generative model of shape textures. Our model serves as a conditional generative model of textured meshes, and also provides a method for inferring them from a single image.

Several recent methods experiment with latent 3D representations (“graphics codes”), largely learned from 2D supervision. In particular, rendering-based generative models have been some of the earliest models able to both infer and generate 3D representations [65]. Liu et al. [47] investigate 3D reconstruction and generation from 2D images using implicit surfaces. PlatonicGAN [23] and IG-GAN [50] utilize a generative model on 2D images, based on rendering voxels. HoloGAN [52] and BlockGAN [53] utilize 3D transformations of feature maps to achieve an implicit 3D representation within a generative model. Works that utilize

learned neural renderers (rather than “hand-designed” ones) may gain an advantage in optimization, but lose some guarantees, such as correct viewpoint invariance [54, 52]. 3D-SDN [78] enables 3D-aware image editing via an inverse graphics approach, utilizing renderers and de-renderers for shape and texture to reconstruct the image. Other approaches also use rendering-based mesh reconstructions in a generative auto-encoding model [21, 22, 34]. These latter approaches are most similar to our work, except that we infer both pose and texture, and utilize unpaired 3D data to impose a prior on the reconstructions.

Another closely related vision technique is the use of 3D information for inter-image correspondence inference. Canonical surface mapping learns a mapping from 2D image pixels to a 3D template, implicitly allowing correspondence computation [40, 39, 73]. The recent work by You et al. [80] consider a 2D-3D-2D cyclic model, to estimate corresponding keypoints in images. Similarly, our model automatically performs correspondence estimation without the need for keypoint data, due to the use of a canonical template and a cycle-consistency requirement to match inferred nodal positions to those in the original input shape.

Lastly, our model may be viewed as a form of 2D-3D modality translation. Such conditional generative modelling techniques impose a cycle-consistency criterion between two domains, and thus provide a powerful framework for domain adaptation [26]. The prototypical example of this family of models is CycleGAN [82], but they do not focus on 3D information. Miyauchi et al. [51] define a cycle-consistent model between an image and its surface normal. Adversarial Inverse Graphics networks [75] are able to perform 2D-3D lifting with unpaired supervision. These approaches do not attempt to completely translate between the 2D and 3D object representation modalities; hence, they do not completely recover 3D shape, pose, and texture as we do. Furthermore, several recent works focus on the synthetic-to-real domain translation problem imposed by 2D image data generated from 3D renderings [61, 60]. Others attempt to induce invariance via domain randomization [71, 70]. In contrast, our work uses distribution matching losses to enforce the generated renders to resemble in-domain images in a data-driven fashion. Finally, we operate on the 3D shape data level (before rendering), rather than altering the resulting renders or simply randomizing the textures.

3. Model

3.1. Overview and Definitions

Our model consists of two main algorithms, which are composed to compute our cycle-consistent training objective, as shown in Fig. 2: the shape-to-image and image-to-shape functions, $f_{S \rightarrow I}$ and $f_{I \rightarrow S}$ (depicted in Fig. 3 and 4, respectively). Given a novel shape, $f_{S \rightarrow I}$ samples a texture and

viewpoint for a novel shape, then renders an image, while $f_{I \rightarrow S}$ performs 3D shape, texture, and viewpoint inference from an input image (inverse graphics). We define latent vectors $\xi_T \sim \mathcal{N}(0, I)$, $\xi_p \sim \mathcal{N}(0, I)$, and v , for the texture (3D appearance), Euclidean pose, and non-rigid shape, respectively. Our shape representation utilizes a template mesh, $S_T = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, written as a tuple of vertices, edges, and faces, which we deform to match either a given input shape or image, through a latent shape vector v . The texture $T \in \mathbb{R}^{|\mathcal{V}| \times 3}$ is simply the colour values per vertex, on the template S_T . The rigid pose transform (mapping object to scene space), $E = (R, t)$, consists of a global rotation $R \in SO(3)$ and translation $t \in \mathbb{R}^3$. The rotation matrix R is computed via the Tait-Bryan angle [11] vector representation $r \in \mathbb{R}^3$, and then converted to R .

Input shapes are assumed to be point clouds $P \in \mathbb{R}^{N_S \times 6}$ in approximate rigid alignment with N_S points, along with their associated surface normals (which we convert to a canonical template representation M), while 2D input images are denoted by I . We use \tilde{v} , \tilde{M} , $\tilde{\xi}_T$, \tilde{E} , and \tilde{I} to denote the shapes (latent and mesh), textures, poses, and renders from a single modality translation application, respectively, and \hat{v} , \hat{M} , $\hat{\xi}_T$, \hat{E} , and \hat{I} for those from a second translation (i.e., reconstructions).

The core of our approach is to enforce that (a) $(\tilde{M}, \tilde{\xi}_T, \tilde{E}) = f_{I \rightarrow S}(I)$ and $\tilde{I} = f_{S \rightarrow I}(M, \xi_T, E)$ produce “realistic” outputs, via adversarial distribution matching, and (b) cycle-consistency is upheld, meaning $I \approx \hat{I} = f_{S \rightarrow I}(f_{I \rightarrow S}(I))$ and $(P, \xi_T, E) \approx (\hat{M}, \hat{\xi}_T, \hat{E}) = f_{I \rightarrow S}(f_{S \rightarrow I}(M, \xi_T, E))$, where ξ_T and E are sampled from their respective priors. We call these two cycles the *vision cycle* (VC) and *graphics cycle* (GC), respectively. For further details, we refer the reader to the appendix (Sec. A).

3.2. Shape-to-Image Translation

Consider an oriented point set, P , which we intend to render into an image (see Fig. 3). From Gaussian priors, we start by sampling a latent texture, ξ_T , and pose, ξ_p , and decode the latter into a Euclidean transform $E = (R, t) = f_p(\xi_p)$. We first map the input, P , into a canonical parametrization, by deforming the template mesh, S_T . This helps ensure that our vertex-wise texture is always computed in the same space, and that there is an approximate correspondence between nodes across input shapes. We thus infer the latent shape vector via $v = f_v(P)$, where f_v is implemented as a standard PointNet [62], able to handle a dynamic number of points. Together, the tuple $\mathfrak{C} = (E, v, \xi_T)$ may be considered the “graphics code” of our model, encompassing shape, appearance, and pose, all in 3D. With some abuse of notation, this code \mathfrak{C} can be viewed as the input to $f_{S \rightarrow I}$ and the output of $f_{I \rightarrow S}$.¹

¹We use the latent vector v and the corresponding mesh M interchangeably to refer to a 3D shape.

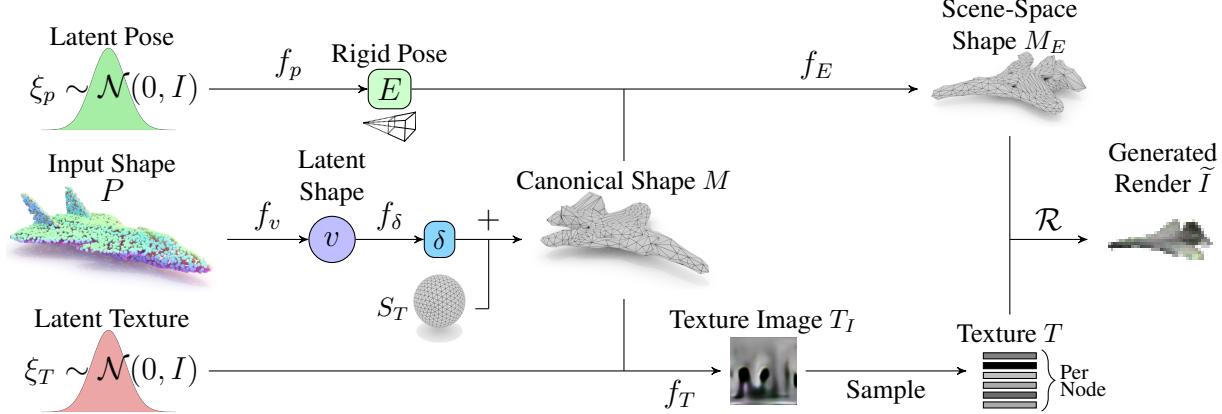


Figure 3. The architecture of our *shape-to-image* translation function, $f_{S \rightarrow I}$. Starting from an input shape P and sampled latent vectors, ξ_T and ξ_p , we first encode P into its latent form v , which is used to obtain the canonical shape, M , via adding the nodal offset δ to the template S_T . Conditioned on M , the latent texture ξ_T is decoded into the UV texture image T_I , which is sampled to obtain the texture T as nodal colours. The rigid pose is independently decoded into a Euclidean transform, E , and applied to M , producing M_E . Finally, M_E and T are differentiably rendered into the image, \tilde{I} .

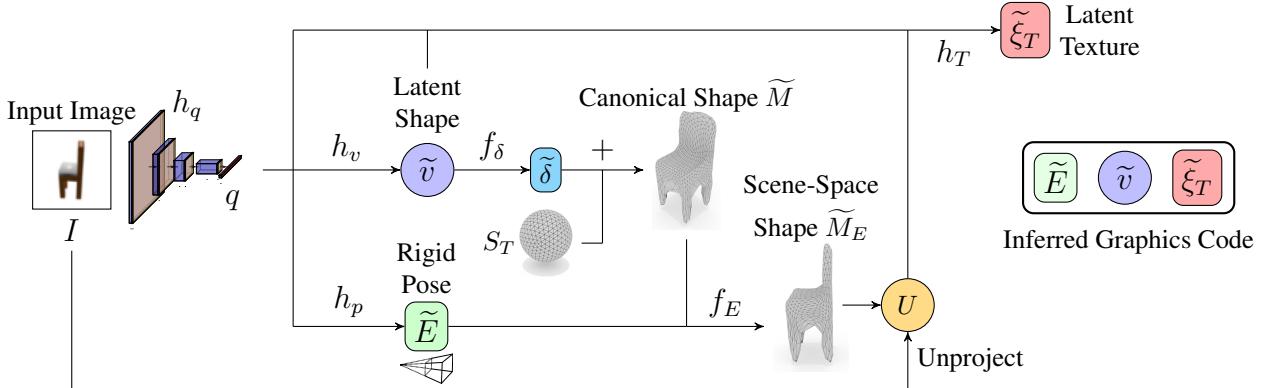


Figure 4. The architecture of the *image-to-shape* translation function $f_{I \rightarrow S}$. Given an input image I , we process it into an encoding q , which is subsequently used to estimate latent shape \tilde{v} and rigid pose \tilde{E} . Then, \tilde{v} is decoded into a nodal perturbation, $\tilde{\delta}$, and applied to the template, S_T , to get \tilde{M} , and \tilde{E} is used to transform \tilde{M} into \tilde{M}_E . Lastly, the pixel values of I are unprojected onto \tilde{M}_E to obtain nodal colours U , which are mapped to ξ_T . The final output is the inferred graphics code $\mathcal{C} = (\tilde{E}, \tilde{v}, \xi_T)$, which can be decoded by $f_{S \rightarrow I}$.

Then, we decode the latent shape v into a set of nodal offsets $\delta = f_\delta(v) \in \mathbb{R}^{|\mathcal{V}| \times 3}$, so that the vertices of the deformed shape may be written as $M = \mathcal{V} + \delta$ (e.g., as in [34]). This canonically parametrized shape, M , is in canonical (object-centred) coordinates, as well. As in an extrinsic camera transform, we apply the Euclidean transform $E = (R, t)$ to obtain the scene-space shape via $M_E = f_E(M, E) = MR + t$, where R and t are applied to every vertex.

Next, we decode the texture, conditionally on the shape: $T = f_T(\xi_T, M)$. Even for a fixed object category, there may be sufficient non-rigid deformation that a given texture will not be applicable to every shape; conditioning on the canonical shape, M , allows the networks to adjust to such variations. We note that the texture decoder, f_T , proceeds by first mapping ξ_T to a texture image T_I (with an upsampling

convolutional network), and then uses the fixed UV texture coordinates of the spherical template to obtain nodal pixel values by bilinear sampling [29] from T_I . Finally, we apply the SoftRas differentiable renderer [46] to generate an image from the textured shape, via $\tilde{I} = \mathcal{R}(M_E, T)$. We use constant ambient lighting and a camera with fixed intrinsics throughout the paper. This complete mapping constitutes the shape-to-image function $\tilde{I} = f_{S \rightarrow I}(f_v(P), \xi_T, E)$, as depicted in Fig. 3.

3.3. Image-to-Shape Translation

Now consider an input image I , from which we wish to extract a textured 3D shape representation (see Fig. 4). We first pass the image through a ResNet-18 [20] to extract a vector image encoding, $q = h_q(I)$. The pose and

latent shape are then estimated from q , as $\tilde{E} = h_p(q)$ and $\tilde{v} = h_v(q)$. As in shape-to-image translation, we deform the template, as $\tilde{M} = \mathcal{V} + f_\delta(\tilde{v})$, and place it in scene coordinates: $\tilde{M}_E = f_E(\tilde{M}, \tilde{E})$.

We next estimate the full 3D texture from the image. Since our inference algorithm is designed to “explain” the observed image as a reprojection of the predicted 3D shape, we can directly detect the colour each vertex *should* be, based on which pixel it is projected to in the original image. Similar to recent work [22, 34, 73], we use this colour information to help infer the latent texture; but, note that these values cannot be used directly, since the generative shape-to-image function $f_{S \rightarrow I}$ expects ξ_T to be Gaussian. Let $U = \mathcal{P}_U(I, \tilde{M}_E) \in \mathbb{R}^{|\mathcal{V}| \times 3}$ be the *pixel unprojection* of the image I onto the transformed shape, \tilde{M}_E . The i^{th} subvector of U is the colour $I(\tilde{x}_i)$, where \tilde{x}_i is the pixel projection of the i^{th} template vertex (from \tilde{M}_E). The inferred latent texture encoding is finally computed via $\tilde{\xi}_T = h_T(q, \tilde{v}, U)$, which can be decoded into a complete texture $\tilde{T} = f_T(\tilde{\xi}_T, \tilde{M})$.

Following prior work involving 3D rigid pose inference [72, 40], we also utilize a set of n_h *pose hypotheses* when performing inference, to avoid getting trapped in local minima. Thus, any call to $f_{I \rightarrow S}$ produces only a single canonical shape \tilde{M} , but n_h rotations \tilde{R}_i , translations \tilde{t}_i , scene-space shapes $\tilde{M}_{E,i}$, textures $\tilde{\xi}_{T,i}$ (due to the pose dependence of the unprojection), and pose probabilities p_i for $i \in [1, n_h]$. In either cycle, these outputs are processed independently until their use in the objective function, similar to [72, 40].

3.4. Cycle-Consistent Loss Objective

Similar to cycle-consistent GANs, our overall loss function consists of adversarial distribution matching and reconstruction consistency losses from the two cycles formed by $f_{I \rightarrow S}$ and $f_{S \rightarrow I}$ (depicted in Fig. 2):

$$\mathcal{L} = \mathcal{L}_G + \mathcal{D}_G + \mathcal{L}_V + \mathcal{D}_V + \mathcal{L}_R, \quad (1)$$

where the reconstruction loss, \mathcal{L}_G , and distribution matching loss, \mathcal{D}_G , form the *graphics cycle* objective (from applying $f_{S \rightarrow I}$, then $f_{I \rightarrow S}$), while \mathcal{L}_V and \mathcal{D}_V do so for the *vision cycle* (applying $f_{I \rightarrow S}$, then $f_{S \rightarrow I}$). The objective term \mathcal{L}_R is comprised of regularization losses. While we expand on each term below, due to space constraints, we refer the reader to the appendix for additional details (Sec. B).

3.4.1 Graphics Cycle Losses

Distribution Matching Loss \mathcal{D}_G . We impose distribution matching losses on the generated render \tilde{I} , texture image T_I (which is sampled to obtain T), and the Euclidean poses, E . The losses on both \tilde{I} and T_I are implemented with convolutional WGAN-GP image critics [2, 16], C_I and C_T respectively. As in [6], we train C_T using inferred textures

\tilde{T} from the vision cycle (VC). Similarly, we use the pose distribution from the VC to impose a loss on E . While one might expect to learn the correct rigid pose distribution from the image adversary, we found this was more stable. Since the Euclidean transform elements, R and t , are low dimensional, we use a Sinkhorn distance [9] (via GeomLoss [12]) to match their distributions. We maintain a buffer of recently inferred \tilde{E} values from the VC, matching each batch from the graphics cycle (GC) to it. We use the L_2 metric for t and the geodesic distance on $SO(3)$, written d_R , for R [27].

Cyclic Consistency Loss \mathcal{L}_G . After computing $\tilde{I} = f_{S \rightarrow I}(f_v(P), \xi_T, E)$, we obtain \tilde{v} , \tilde{M} , \tilde{E} , and $\tilde{\xi}_T$ from $f_{I \rightarrow S}(\tilde{I})$. Consistency in shape, texture, and pose is enforced by matching E and \tilde{E} , ξ_T and $\tilde{\xi}_T$, and (v, M) and (\tilde{v}, \tilde{M}) , respectively, all as vectors (except for R and \tilde{R} , which use d_R). We also use a metric between the input P and the canonically deformed template shape M , consisting of a Chamfer and a normals-matching loss, as in [14].

3.4.2 Vision Cycle Losses

Distribution Matching Loss \mathcal{D}_V . There are two distribution-matching losses in the VC. The first is on the inferred shape (\tilde{v}, \tilde{M}) , which we implement as a vector critic C_s . This loss encourages the canonical shape \tilde{M} , inferred from the image I , to look like a deformed template, obtained by encoding a *real* 3D shape, P , in the GC. The second is on the inferred latent texture, $\tilde{\xi}_T$, which we enforce to be Gaussian, consistent with our sampling in the GC. Since ξ_T is relatively low-dimensional, we follow recent work in generative models and apply the sliced Wasserstein distance (SWD) [10, 63] to push $\tilde{\xi}_T$ to be normally distributed, due to its stability and dearth of (hyper-)parameters. As noted in Sec. 3.4.1, we do not use a critic on \tilde{E} , instead relying on the rigid poses inferred in the VC to be realistic.

We remark here that all adversarial critics (i.e., those applied on the inferred shapes, $C_s(\tilde{v}, \tilde{M})$, generated textures, $C_T(\tilde{T}_I)$, and sampled renders, $C_I(\tilde{I})$) use the WGAN-GP loss formulation [16], with an additional “drift” penalty [35], to prevent losses from growing too large and overwhelming other generator terms.

Cyclic Consistency Loss \mathcal{L}_V . The only reconstruction loss in the VC is the image-to-image distance between I and the reconstructed render $\hat{I} = f_{S \rightarrow I}(f_{I \rightarrow S}(I))$. We use a combination of an L_1 pixelwise loss and a perceptual metric, computed using the intermediate feature maps of the image critic C_I (as in, e.g., [42, 77]). The loss is weighted as an expectation over the pose hypothesis probabilities, p_i .

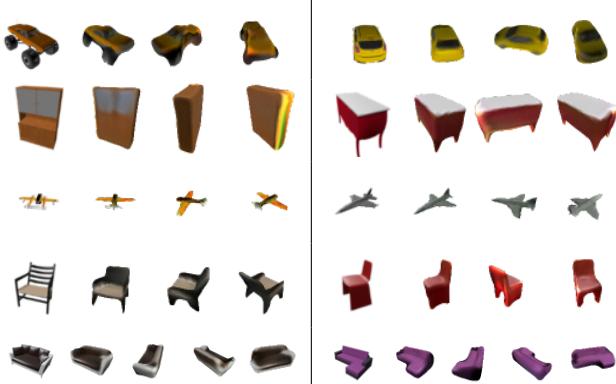


Figure 5. Qualitative visualization of 3D reconstruction results. Per set: first image is the input I , second is the re-rendered reconstruction \hat{I} , and the last two insets are different views of the output. Note that our masked input images are given a black background; hence the network chooses to fill “holes” with black (fourth row, left).

3.4.3 Regularization Losses

As in most learning models that operate on meshes, we apply geometric regularizations to prevent pathological shape outputs (e.g., rampant self-intersection [22] and “flying” vertices [76]). In particular, we follow prior work [36, 76, 46, 14] and apply a Laplacian, flatness, and edge length losses to the canonical meshes M and \tilde{M} . We add an additional term for δ and $\tilde{\delta}$, penalizing the per-face variance of the nodal perturbations.

We also regularize the exploration of the Euclidean pose hypotheses. As in [40], we encourage diversity in the rotations by maximizing the expected pairwise distance between the rotational hypotheses, via $L_{RD} = \sum_{i,j} p_i p_j d_R(R_i, R_j)$. For t and ξ_T in the VC, we penalize the distance of each hypothesis to the “best” (highest probability) vector, instead. This helps decrease “drift” in the inferred translation \tilde{t} , which reduces instability (e.g., due to the mesh leaving the view frustum). For the latent texture $\tilde{\xi}_T$, we encourage consistent textures regardless of the estimated pose.

Finally, while texture may be conditional on the shape, it should be *independent of the pose*, so we encourage their disentanglement by adversarial means. This is necessitated by the unprojection step, which introduces a reliance of the inferred latent texture, $\tilde{\xi}_T$, on the predicted pose, \tilde{E} . Hence, we train a small convolutional adversary network A_R to infer the (best) rotation R_b from the (best) texture image $T_{I,b}$; i.e., minimize $d_R(R_b, A_R(T_{I,b}))$. Our model, $f_{S \rightarrow I}$ and $f_{I \rightarrow S}$, is then trained to maximize the error of this regressor (only for the VC). Separately, we additionally regularize the latent shape space v with an SWD loss (as used for ξ_T), which helps the VC satisfy the shape critic C_s , and is also used to help *ex-post* distribution fitting to v (see Sec. 4.2).

Category	F-score (τ) \uparrow			F-score (2τ) \uparrow			Chamfer \downarrow		
	3DR	Ours	P2M	3DR	Ours	P2M	3DR	Ours	P2M
plane	41.5	70.0	71.1	63.2	81.9	81.4	0.90	0.37	0.48
cabinet	49.9	46.7	60.4	64.8	60.6	77.2	0.74	0.90	0.38
car	37.8	58.2	67.9	54.8	70.7	84.2	0.85	0.56	0.27
chair	40.2	35.0	54.4	55.2	46.0	70.4	1.43	2.60	0.61
sofa	40.0	39.6	51.9	53.4	52.7	69.8	1.14	1.26	0.49

Table 1. Reconstruction performance on the ShapeNet-based test set. Metrics: F-score (%) at two thresholds, where $\tau = 10^{-4}$ and larger is better (\uparrow), and Chamfer distance ($\times 1000$), where smaller is better (\downarrow). Values for 3D-R2N2 (3DR) and Pixel2Mesh (P2M) are taken from [76]. While lower than the supervised counterparts, performance of our model still comes close, even with our more complex representational requirements; in particular, we are able to either exceed or come within 10% of the F-score of 3D-R2N2 (see Sec. 4.1).

4. Experimental Results

For all experiments, we use the shape data from ShapeNet [5] (without textures) and the rendered image data from Choy et al. [7]. We remove internal mesh structure by voxelization and marching cubes [49], using Kaolin [30]. At no point is the relation between shapes and renders used by our model. All models are implemented in PyTorch [58], and optimized with Adam [37]. Hyper-parameters are tuned for visual quality on the training set, and are the same across categories, except for the planes class (see Sec. C.1.2). We refer the reader to the appendix for additional experimental details (Sec. C) and results (Sec. D).

4.1. 3D Reconstruction

As a by-product of learning our weakly supervised modality translation model, we obtain the ability to perform 3D shape reconstruction from 2D images, via the first half of the VC (i.e., the image-to-shape function $f_{I \rightarrow S}$). Qualitative results are shown in Fig. 5. The network obtains the overall shape and captures visible appearance details fairly well (e.g., car windows). However, it has difficulty texturing occluded parts (e.g., the backs of the chairs and cabinets). Other works mitigate occlusion issues with symmetry constraints [73, 34], but these make assumptions on the object. Issues with the fixed topology of the template are also visible in the chair reconstructions, as well as some of the planes (left inset). Overall, however, the model is able to infer a reasonable shape and texture, despite the weak supervision.

For quantitative comparison (see Table 1), we use the same metrics and held-out testing set as Pixel2Mesh [76]. Since Pixel2Mesh is trained with the 3D ground truth in camera coordinates per input image, whereas our model decides on its own camera coordinates based only on explaining the image evidence, we align the 3D reconstructions in position and scale for fair comparison. This is similar to a Procrustes

Category	IS \uparrow				FID \downarrow				KID \downarrow			
	VAE	Ours	OursU	GAN	VAE	Ours	OursU	GAN	VAE	Ours	OursU	GAN
plane	2.36	2.98	2.86	3.39	92.3	58.9	59.4	11.5	8.83	5.00	5.16	0.69
cabinet	3.64	3.16	3.20	3.80	176.8	131.6	136.6	73.8	15.73	10.94	11.55	6.53
car	2.98	3.12	3.10	3.84	194.3	119.6	121.0	9.1	18.61	10.37	10.51	0.491
chair	4.10	3.96	3.90	5.21	119.9	108.8	110.8	45.8	10.23	9.27	9.43	4.00
sofa	3.73	3.86	3.79	4.66	171.9	94.3	98.1	27.7	14.89	7.07	7.44	1.78

Table 2. Performance on generative modelling. Metrics: Inception Score (IS), Frechet Inception Distance (FID), and Kernel Inception Distance (KID; $\times 100$). OursU denotes *unconditional* generation via *ex-post* fitting. We perform similarly to the VAE, despite our constrained representation (see Sec. 4.2).



Figure 6. Example generations from our shape-to-image function $f_{S \rightarrow I}$ with randomly sampled texture and pose.

analysis [3] (centering and scalar scale matching), but without performing the rotation matching, since we expect our model to approximately output the correct orientation. In particular, we sample points from our inferred mesh \widetilde{M}_E , transform it into camera coordinates, translate it (and the ground truth shape) to the origin based on the mean point value, and then scale our point set by an analytically computed scalar factor s , which corrects for the difference in (a) the distance to the camera along the optical axis and (b) the projective intrinsics of our renderer versus those used to render the ground truth data. We note that this is important for quantitative evaluation, since point cloud metrics are sensitive to small differences in scale and rigid misalignments. As in [76], we utilize the Chamfer distance and F-score (precision and recall between point sets at a fixed threshold) to evaluate the method.

Results are shown in Table 1. While our method does not perform as well as the Pixel2Mesh model, we are able to perform comparably to the voxel-based 3D-R2N2 model (either exceeding its performance or coming within a 10% margin by F-score), despite it being fully supervised with paired data. Furthermore, we note that our model is less specialized than those we compare to, as it must learn additional capabilities (e.g., texture estimation, generative modelling); the capacity used by these additional tasks is expected to reduce performance (as is the weaker supervision).

4.2. Generative Modelling

We next investigate the performance of our network as a generative model, using the first half of the GC (via the shape-to-image function $f_{S \rightarrow I}$) to render an input shape P based on a random pose ξ_p and texture ξ_T . Qualitative example generations are shown in Fig. 6. We see that the model is able to output a diversity of shapes (especially in the chairs and sofas), as well as acceptable textures (e.g., the details on the cars). While point sets, P , are given as inputs, we remark that fitting a surface mesh to a point cloud is itself a non-trivial problem [19], which our shape encoder f_v must solve with the template, S_T , and input, P . In addition, we note that white or grey monocoloured textures are disproportionately common, particularly for the plane and sofa datasets, which is reflected in our generated distribution as well.

However, since our model is actually a *conditional* generative model, we should remove this dependency for fairer comparison to methods that do not have access to a shape input. Thus, to make our model *unconditional*, we follow work in generative autoencoders [13, 1] and perform *ex-post* density estimation to the regularized latent shape space v . In particular, we fit a Gaussian mixture model (GMM) Q_v to the approximate aggregate posterior of the output of f_v . In the context of generative models, this GMM serves as the latent prior over shapes. We therefore obtain an unconditional model over textured shapes by sampling $v \sim Q_v$ and $\xi_T, \xi_p \sim \mathcal{N}(0, I)$.

For comparison, we consider two standard generative models on 2D images: a GAN [15] and a VAE [38, 66]. While a GAN is a useful gold-standard baseline, we do not expect to outperform it on image generation, as it (i) does not have to learn an inference model, (ii) can manipulate pixels directly (as opposed to colouring vertices, followed by rasterization), and (iii) does not attempt to model 3D information. An image VAE, on the other hand, has to (i) learn an encoder and (ii) ensure that the inferred latent variables are approximately Gaussian. This is analogous to our model’s texture representation, wherein the latent $\widetilde{\xi}_T$ is inferred from the input image and optimized to follow a Gaussian distribution. Hence, though it still has the advantage of operating on pixels and not having to extract or generate 3D shape and pose, an image VAE may be a more appropriate baseline than a GAN. For implementations, we use a WGAN-GP [16] and a standard convolutional VAE, from public implementations

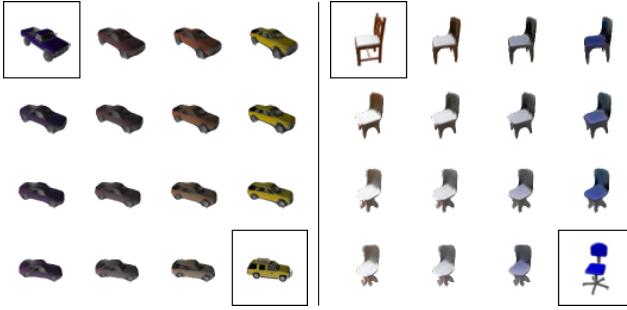


Figure 7. Decoded renderings of latent space interpolations. Per inset: boxed corner images represent start and end points (real images), vertical axis corresponds to interpolation in latent shape (v) and Euclidean pose (E), while the horizontal axis represents interpolation of the latent texture (ξ_T). Non-boxed opposing corners (upper-right and lower-left) may be viewed as *texture transfers*.

in [43] and [69], respectively. We show quantitative results with these models in Table 2, using standard reference-free image quality metrics from the GAN literature: Frechet Inception Distance (FID) [24], Inception Score (IS) [67], and Kernel Inception Distance (KID) [4], computed via [55]. We use the train-test split from [76] for evaluation. Unsurprisingly, the GAN scores the best in all cases, but we find that our model generates images of similar quality as the VAE. Our goal is not to show that there does not exist a VAE that could generate images of higher quality; rather, we wish to demonstrate that our network is able to perform at a similar level to a standard pixel-level generative model, despite the additional constraints on our model. Finally, we remark that there is a fundamental domain gap between the output of our generative renderer (which has limited mesh resolution and a soft rasterization step) and that of the process that created the input pixel data, which limits our performance. For future work, we remark that texture quality could be improved by more sophisticated representations (e.g., texture fields [56]).

4.3. Latent Textured Shape Representation

We highlight two additional applications of our network to representation learning: (i) latent shape-pose-texture disentanglement and (ii) unsupervised correspondence discovery. The first task concerns the interpretable nature of our latent “graphics code” representation, showing that we can manipulate the 3D pose, intrinsic shape, and texture of an object, independently. This is showcased in Fig. 7, where we interpolate through latent shape and pose, separately from texture. The second task considers the natural geometric correspondence that occurs due to the use of a canonical template, S_T , which enables the network to assign a meaning to each vertex that is consistent across shapes. The order dependent cyclic shape loss in the GC (M -to- \widehat{M}) further encourages consistency, as it demands inference of the *same* position for each vertex, across the rendering process. Thus,

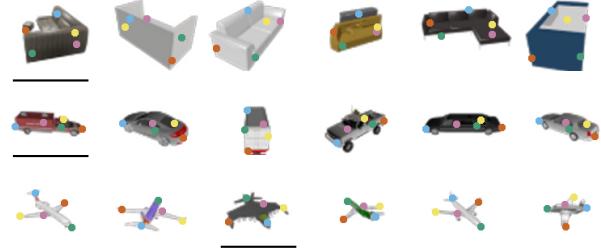


Figure 8. Unsupervised corresponding keypoint identification. Each row shows a set of real input images from our dataset, along with several marked keypoints in correspondence across images (obtained by projecting the inferred output shape onto each one). The same colour of point denotes the same template vertex index across images. Note that we show the keypoint even if the originating vertex is occluded in the image. Underlines highlight shapes that the model incorrectly inferred “backwards”.

by looking at the projected positions of specific template nodes (i.e., inferring \widehat{M}_E from the input, and marking where the chosen vertices project to in I), we obtain approximately corresponding points on the original images, similar to an unsupervised keypoint discovery method. We show examples of this in Fig. 8, where each colour point corresponds to the *same* template node index. The results show that the network is often able to identify “corresponding” points across images, despite changes in pose, shape, and appearance. They also reveal areas of difficulty for the network (e.g., confusing the front and back of approximately symmetric objects, as in the underlined insets).

5. Conclusion

We have presented a unified learning framework for *modality translation* between 2D images and 3D textured shape. Our method combines a shape-to-image mapping, $f_{S \rightarrow I}$, which performs generative rendering, with an image-to-shape mapping, $f_{I \rightarrow S}$, that infers a latent 3D representation from a single image. We learn these functions by optimizing over two cyclic consistency losses that alternate these mappings: the *graphics* and *vision* cycles. Our model works directly on triangle meshes, and is trained with weak supervision using *unpaired* data. We demonstrated its potential on single-image 3D reconstruction, textured shape generation, and latent representation learning. However, our method currently does not handle images with backgrounds, is topologically constrained in 3D shape by its use of a deformable template based on nodal offsets (e.g., Fig. 7, right inset), and does not separate lighting from texture. Nevertheless, we expect our model to be useful in helping machine learning models make better use of data with less supervision, as well as in obtaining generative representations with interpretable 3D latent factors.

References

- [1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3D point clouds. In *International Conference on Machine Learning (ICML)*, pages 40–49, 10–15 Jul 2018. 7
- [2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 214–223, 2017. 5, 13, 14
- [3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (5):698–700, 1987. 7, 16
- [4] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. *arXiv preprint arXiv:1801.01401*, 2018. 8, 16
- [5] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An information-rich 3D model repository. 2015. 6
- [6] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3D objects with an interpolation-based differentiable renderer. In *Neural Information Processing Systems (NeurIPS)*, pages 9605–9616, 2019. 2, 5
- [7] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3DR2N2: A unified approach for single and multi-view 3D object reconstruction. In *European Conference on Computer Vision (ECCV)*, 2016. 6, 16
- [8] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015. 11
- [9] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Neural Information Processing Systems (NeurIPS)*, pages 2292–2300, 2013. 5, 13
- [10] I. Deshpande, Z. Zhang, and A. G. Schwing. Generative modeling using the sliced Wasserstein distance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3483–3491, 2018. 5, 13
- [11] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15–16):1–35, 2006. 3
- [12] J. Feydy, T. Séjourné, F.-X. Vialard, S.-I. Amari, A. Trouvé, and G. Peyré. Interpolating between optimal transport and MMD using Sinkhorn divergences. *arXiv preprint arXiv:1810.08278*, 2018. 5
- [13] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. J. Black, and B. Schölkopf. From variational to deterministic autoencoders. 2019. 7
- [14] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 9785–9795, 2019. 2, 5, 6, 13, 14
- [15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014. 2, 7, 16
- [16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. In *Neural Information Processing Systems (NeurIPS)*, pages 5767–5777, 2017. 5, 7, 12, 13, 14
- [17] J. Gwak, C. B. Choy, M. Chandraker, A. Garg, and S. Savarese. Weakly supervised 3D reconstruction with adversarial constraint. In *International Conference on 3D Vision (3DV)*, pages 263–272. IEEE, 2017. 2
- [18] X. Han, H. Laga, and M. Bennamoun. Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2019. 2
- [19] R. Hanocka, G. Metzger, R. Giryes, and D. Cohen-Or. Point2Mesh: A self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084*, 2020. 7
- [20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4, 12, 14
- [21] P. Henderson and V. Ferrari. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, pages 1–20, 2019. 3
- [22] P. Henderson, V. Tsiminiaki, and C. H. Lampert. Leveraging 2D data to learn textured 3D mesh generation. *arXiv preprint arXiv:2004.04180*, 2020. 2, 3, 5, 6
- [23] P. Henzler, N. Mitra, and T. Ritschel. Escaping Plato’s cave using adversarial training: 3D shape from unstructured 2D image collections. *arXiv preprint arXiv:1811.11606*, 2018. 2
- [24] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems (NeurIPS)*, pages 6626–6637, 2017. 8, 16
- [25] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations (ICLR)*, 2017. 16
- [26] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017. 3
- [27] D. Q. Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009. 5
- [28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 11
- [29] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Neural Information Processing Systems (NeurIPS)*, pages 2017–2025, 2015. 4, 12
- [30] K. M. Jatarellabhu, E. Smith, J.-F. Lafleche, C. F. Tsang, A. Rozantsev, W. Chen, and T. Xiang. Kaolin: A PyTorch library for accelerating 3D deep learning research. *arXiv preprint arXiv:1911.05063*, 2019. 6, 16
- [31] S. Jenni and P. Favaro. On stabilizing generative adversarial training with noise. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 14

- [32] L. Jiang, S. Shi, X. Qi, and J. Jia. Gal: Geometric adversarial loss for single-view 3D-object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 802–816, 2018. 2
- [33] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7122–7131, 2018. 2
- [34] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 2, 3, 4, 5, 6
- [35] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 5, 14
- [36] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916, 2018. 2, 6
- [37] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6, 14, 15
- [38] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. 7, 16
- [39] N. Kulkarni, A. Gupta, D. F. Fouhey, and S. Tulsiani. Articulation-aware canonical surface mapping. *arXiv preprint arXiv:2004.00614*, 2020. 3
- [40] N. Kulkarni, A. Gupta, and S. Tulsiani. Canonical surface mapping via geometric cycle consistency. In *International Conference on Computer Vision (ICCV)*, pages 2202–2211, 2019. 3, 5, 6, 14
- [41] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Neural Information Processing Systems (NeurIPS)*, pages 2539–2547, 2015. 1
- [42] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Computer Vision (ICCV)*, 2016. 5, 13
- [43] K. S. Lee and C. Town. Mimicry: Towards the reproducibility of GAN research. *CVPR Workshop on AI for Content Creation*, 2020. 8, 12, 16
- [44] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 2
- [45] S. Liu, W. Chen, T. Li, and H. Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv preprint arXiv:1901.05567*, 2019. 14
- [46] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *International Conference on Computer Vision (ICCV)*, pages 7708–7717, 2019. 2, 4, 6, 12, 14
- [47] S. Liu, S. Saito, W. Chen, and H. Li. Learning to infer implicit surfaces without 3D supervision. In *Neural Information Processing Systems (NeurIPS)*, pages 8293–8304, 2019. 2
- [48] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):1–16, 2015. 2
- [49] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH computer graphics*, 21(4):163–169, 1987. 6
- [50] S. Lunz, Y. Li, A. Fitzgibbon, and N. Kushman. Inverse graphics GAN: Learning to generate 3D shapes from unstructured 2D data. *arXiv preprint arXiv:2002.12674*, 2020. 2
- [51] Y. Miyauchi, Y. Sugano, and Y. Matsushita. Shape-conditioned image generation by learning latent appearance representation from unpaired data. In *Asian Conference on Computer Vision*, pages 438–453. Springer, 2018. 3
- [52] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. In *International Conference on Computer Vision (ICCV)*, pages 7588–7597, 2019. 2, 3
- [53] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra. BlockGAN: Learning 3D object-aware scene representations from unlabelled images. *arXiv preprint arXiv:2002.08988*, 2020. 2
- [54] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. In *Neural Information Processing Systems (NeurIPS)*, pages 7891–7901, 2018. 3
- [55] A. Obukhov, mseitzer, willylulu, S. Zhydenko, J. Kyl, and E. Y.-J. Lin. toshas/torch-fidelity: Version 0.2.0, May 2020. 8
- [56] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger. Texture fields: Learning texture representations in function space. In *International Conference on Computer Vision (ICCV)*, pages 4531–4540, 2019. 2, 8
- [57] J. Pan, J. Li, X. Han, and K. Jia. Residual MeshNet: Learning to deform meshes for single-view 3D reconstruction. In *International Conference on 3D Vision (3DV)*, pages 719–727. IEEE, 2018. 2
- [58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché Buc, E. Fox, and R. Garnett, editors, *Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6, 15
- [59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 16
- [60] X. Peng and K. Saenko. Synthetic to real adaptation with generative correlation alignment networks. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, pages 1982–1991, 2018. 3
- [61] X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko. Syn2Real: A new benchmark for synthetic-to-real visual domain adaptation. *arXiv preprint arXiv:1806.09755*, 2018. 3
- [62] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmenta-

- tion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 3, 12
- [63] J. Rabin, G. Peyré, J. Delon, and M. Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 435–446, 2011. 5, 13
- [64] A. Raj, C. Ham, C. Barnes, V. Kim, J. Lu, and J. Hays. Learning to generate textures on 3D meshes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 32–38, 2019. 2
- [65] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3D structure from images. In *Neural Information Processing Systems (NeurIPS)*, pages 4996–5004, 2016. 2
- [66] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic back-propagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014. 7, 16
- [67] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *Neural Information Processing Systems (NeurIPS)*, pages 2234–2242, 2016. 8, 16
- [68] R. Sawhney and K. Crane. Boundary first flattening. *ACM Transactions on Graphics (ToG)*, 37(1):1–14, 2017. 15
- [69] A. Subramanian. PyTorch-VAE. <https://github.com/AntixK/PyTorch-VAE>, 2020. 8, 16
- [70] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017. 1, 3, 15
- [71] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977, 2018. 1, 3, 15
- [72] S. Tulsiani, A. A. Efros, and J. Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 5
- [73] S. Tulsiani, N. Kulkarni, and A. Gupta. Implicit mesh reconstruction from unannotated image collections. *arXiv preprint arXiv:2007.08504*, 2020. 3, 5, 6
- [74] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2626–2634, 2017. 2
- [75] H.-Y. F. Tung, A. W. Harley, W. Seto, and K. Fragiadaki. Adversarial inverse graphics networks: Learning 2D-to-3D lifting and image-to-image translation from unpaired supervision. In *International Conference on Computer Vision (ICCV)*, pages 4364–4372. IEEE, 2017. 1, 3
- [76] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 2, 6, 7, 8, 14, 15, 16
- [77] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8798–8807, 2018. 5, 13
- [78] S. Yao, T. M. Hsu, J.-Y. Zhu, J. Wu, A. Torralba, B. Freeman, and J. Tenenbaum. 3D-aware scene manipulation via inverse graphics. In *Neural Information Processing Systems (NeurIPS)*, pages 1887–1898, 2018. 1, 3
- [79] I. Yildirim, T. D. Kulkarni, W. A. Freiwald, and J. B. Tenenbaum. Efficient analysis-by-synthesis in vision: A computational framework, behavioral tests, and comparison with neural representations. In *Proceedings of the Conference of the Cognitive Science Society*, 2015. 1
- [80] Y. You, C. Li, Y. Lou, Z. Cheng, L. Ma, C. Lu, and W. Wang. Semantic correspondence via 2D-3D-2D cycle. *arXiv preprint arXiv:2004.09061*, 2020. 3
- [81] A. Yuille and D. Kersten. Vision as Bayesian inference: Analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006. 1
- [82] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017. 2, 3
- [83] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. Tenenbaum, and B. Freeman. Visual object networks: Image generation with disentangled 3D representations. In *Neural Information Processing Systems (NeurIPS)*, pages 118–129, 2018. 2
- [84] S. Zuffi, A. Kanazawa, and M. J. Black. Lions and tigers and bears: Capturing non-rigid, 3D, articulated shape from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3955–3963, 2018. 2
- [85] S. Zuffi, A. Kanazawa, D. W. Jacobs, and M. J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6365–6373, 2017. 2

A. Model Details

We first provide details on the architectures of the component networks defining our modality translation functions. We set the number of pose hypotheses to $n_h = 4$, and the various encoding dimensionalities as $\dim(v) = 64$, $\dim(q) = 512$, $\dim(\xi_p) = 16$, and $\dim(\xi_T) = 128$.

Notationally, we write $LBA(n_1, n_2, \dots)$ to mean a multilayer perceptron with hidden layers of size n_1, n_2, \dots , ending in a linear layer. Each layer consists of a fully connected linear layer (L), followed by batch normalization (B) [28], and non-linear ELU activation (A) [8]. We append a BA to the end, if we include a B and A after the final linear layer. We also use the notation $f : a \rightarrow b$ to remind the reader of the input and output variables of the mapping f . The same variable definitions are used as in the main text.

Model component architectures are as follows:

- The pose decoder $f_p : \xi_p \rightarrow (r, t)$ is given by LBA(64), but note the additional step converting from Tait-Bryan angles $r \in \mathbb{R}^3$ to a rotation matrix $R \in SO(3)$, so that the Euclidean transform for rigid pose is given by $E = (R, t)$.
- The shape encoder $f_v : P \rightarrow v$ is implemented as a standard PointNet [62], without spatial transformers. Hidden layers have sizes 64, 128, 1024, 512, and 512.
- The nodal perturbation decoder $f_\delta : v \rightarrow \delta$ is written LBA(600, 900, 1200).
- Template mesh S_T : we use a sphere with $|\mathcal{V}| = 1002$ and $|\mathcal{F}| = 2000$.
- The image encoder $h_q : I \rightarrow q$ is a standard ResNet-18 backbone [20].
- The latent shape inferrer $h_v : q \rightarrow \tilde{v}$ is a single fully connected layer.
- The pose inferrer $h_p : q \rightarrow \tilde{E}$ is given by LBA($12 * n_h$) BA, but note the additional step converting from Tait-Bryan angles to a rotation matrix, and the presence of hypotheses.
- The pose hypothesis probability estimator $h_e : q \rightarrow H$, where $H = (p_1, \dots, p_{n_h})$, is given by LBA($32 * n_h$)-softmax.
- The latent texture inference network $h_T : (q, \tilde{v}, U) \rightarrow \tilde{\xi}_T$ is decomposed into three subnetworks:
 1. A latent shape preprocessor $g_v : v \rightarrow z_v$, where $\dim(z_v) = 32$, is given by LBA(128) BA.
 2. An unprojection preprocessor $g_u : U \rightarrow z_u$, where $\dim(z_u) = 512$, is given by LBA(512) BA.
 3. The final sub-network for latent texture inference, $g_z : (q, z_v, z_u) \rightarrow \tilde{\xi}_T$, is written LBA($3 * Dz, 2 * Dz$), where $Dz = 32 + 512 + 512$.
- The texture decoder $f_T : (\xi_T, M) \rightarrow T$ consists of three sub-operations:
 1. The mesh preprocessor $g_M : M \rightarrow z_M$, where $\dim(z_M) = 400$, is written LBA(800).
 2. The texture image generator $G_{T_I} : (\xi_T, z_M) \rightarrow T_I$ is a standard image generation network [16], utilizing transposed convolutions in residual blocks. Specifically, it consists of a linear layer, (to dimensionality $4^2 \times 1024$, followed by reshaping into a 4×4 image. Four residual generator blocks are then applied, which halve the channel dimension and double the spatial resolution, and a final 2D conv layer (after a BatchNorm and ReLU) is applied, followed by tanh. The texture image T_I is a 64×64 RGB image. (See implementation in [43], under 64×64 WGAN generators).
 3. We finally obtain the nodal colours from the texture image via bilinearly interpolated grid sampling [29].
- The SoftRas differentiable renderer [46] is used for $\mathcal{R} : (T, M_E) \rightarrow I$. We set the probabilistic spatial sharpness to $\sigma = 10^{-5}$ and the colour aggregation sharpness to $\gamma = 10^{-4}$.

A.1. Detachments

We found it useful to make several judicious detachments within the computation graph (i.e., prevent gradient back-propagation flow along particular connections). This includes the connection between M and f_T , v and h_T , and E and f_E . The former two prevent the network from using node positions to store texture information, while the latter mitigates instability from the adversarial gradients flowing into the pose generator.

A.2. Timing

We note that a single complete training iteration for our model takes ~ 3.3 s, including forward and backward passes for both cycles (graphics and vision), as well as for all critics and adversaries. Our model is trained for 10^5 iterations (see Sec. C.1.1 for training regime details), which takes ~ 90 hours on a single NVIDIA Tesla V100 GPU. The vision cycle forward pass costs ~ 1.2 s, while that of graphics cycle costs ~ 0.9 s. The costliest critics are the Sinkhorn-based pose matcher (~ 0.1 s) and the generated render adversary C_I (~ 0.09 s). For the translations alone, running shape-to-image translation takes ~ 0.25 s, while image-to-shape translation takes ~ 0.4 s. These numbers utilized a single NVIDIA Tesla V100 GPU and a batch size of 64.

B. Loss Details

We now expand on the loss terms described in the main text of the paper. Recall that the complete objective function is given by

$$\mathcal{L} = \mathcal{L}_G + \mathcal{D}_G + \mathcal{L}_V + \mathcal{D}_V + \mathcal{L}_R. \quad (2)$$

B.1. Graphics Cycle Losses

B.1.1 Distribution Matching Loss \mathcal{D}_G

The distribution matching loss for the graphics cycle, \mathcal{D}_G , is written as

$$\mathcal{D}_G = -\gamma_{g,I} C_I(\tilde{I}) - \gamma_{g,T} C_T(\tilde{T}_I) + \gamma_{g,E} \mathcal{S}_D[\mathcal{B}_{\tilde{E}} || E], \quad (3)$$

where the first two terms (using C_I and C_T) are WGAN generator losses [2, 16] (see Sec. B.4), and the final term is the Sinkhorn distance [9] between the pose buffer $\mathcal{B}_{\tilde{E}}$ of recently inferred (best hypothesis) poses from the vision cycle (buffer size 128) and the currently sampled batch of poses, simply denoted here as $E = (R, t)$. This latter term is decomposed into two terms, for the rotations and translations, respectively:

$$\mathcal{S}_D[\mathcal{B}_{\tilde{E}} \parallel E] = \mathcal{S}_D[\mathcal{B}_{\tilde{R}} \parallel R] + \mathcal{S}_D[\mathcal{B}_{\tilde{t}} \parallel t]. \quad (4)$$

We use the L_2 ground metric for t and the geodesic distance on $SO(3)$, denoted d_R , for R :

$$d_R(R_1, R_2) = \arccos([\text{tr}(R_1 R_2^T) - 1]/2). \quad (5)$$

We set the weights as $\gamma_{g,I} = 0.2$, $\gamma_{g,T} = 0.2$, and $\gamma_{g,E} = 10$.

B.1.2 Cyclic Consistency Loss \mathcal{L}_G

Recall that the template mesh is given by $S_T = (\mathcal{V}, \mathcal{E}, \mathcal{F})$. Let $M = (\mathcal{V}_M, \mathcal{E}, \mathcal{F})$ be the canonical deformation of S_T , which alters neither the edges nor the faces, where $\mathcal{V}_M \in \mathbb{R}^{|\mathcal{V}| \times 3}$ is the matrix of nodal coordinates. The cyclic consistency loss for the graphics cycle, \mathcal{L}_G , is then written as

$$\mathcal{L}_G = \frac{\gamma_{g,v}}{\dim(v)} \|v - \hat{v}\|_1 + \frac{\gamma_{g,M}}{|\mathcal{V}|} \|\mathcal{V}_M - \hat{\mathcal{V}}_M\|_F^2 + \quad (6)$$

$$\frac{\gamma_{g,t}}{3} \|t - \hat{t}\|_2^2 + \frac{\gamma_{g,\xi_T}}{\dim(\xi_T)} \|\xi_T - \hat{\xi}_T\|_2^2 + \quad (7)$$

$$\frac{\gamma_{g,R}}{\pi} \sum_{i=1}^{n_h} \hat{p}_i d_R(R, \hat{R}_i) + \mathcal{D}(P, M), \quad (8)$$

where the input oriented point set $P = (X, N)$ is written as a tuple of 3D point coordinates X and normals N . The final term, matching the inferred canonical template mesh M to the input shape P is written as

$$\mathcal{D}(P, M) = \gamma_{g,X} D_C(X_M, X) - \gamma_{g,N} |N \cdot N_M|, \quad (9)$$

where X_M and N_M are uniformly sampled from M , and D_C is the Chamfer distance. We denote $|N \cdot N_M|$ to mean the average per-point dot product between a surface normal $n \in N$ at a point $x \in X$, and the surface normal $n_M \in N_M$, associated to the point $x_M \in X_M$ that is closest to x . In other words, the first term matches the point set and the second matches the normals, as in [14]. We perform this loss on the vertices, as well as the samples, to ensure each template vertex is given a signal at every step. We further remark that \hat{p} refers to the pose hypothesis probabilities computed when applying the modality translation $f_{I \rightarrow S}$, in the second half of the graphics cycle.

We set the weights as $\gamma_{g,v} = 5$, $\gamma_{g,M} = 1000$, $\gamma_{g,t} = 10$, $\gamma_{g,\xi_T} = 5$, $\gamma_{g,R} = 5$, $\gamma_{g,X} = 2000$, and $\gamma_{g,N} = 5$.

B.2. Vision Cycle Losses

B.2.1 Distribution Matching Loss \mathcal{D}_V

The distribution matching loss for the vision cycle, \mathcal{D}_V , consists of two loss terms:

$$\mathcal{D}_V = -\gamma_{v,S} C_s(\tilde{v}, \widetilde{M}) + \gamma_{v,\xi_T} \mathfrak{D}_{\text{SWD}}[\tilde{\xi}_T \parallel \xi_T], \quad (10)$$

where C_s is the WGAN shape critic loss (see Sec. B.4) and $\mathfrak{D}_{\text{SWD}}$ is the sliced Wasserstein distance (SWD) [10, 63] loss between the inferred $\tilde{\xi}_T$ and the normally distributed latent textures $\xi_T \sim \mathcal{N}(0, I)$ (using 150 projections). We remark that the loss on $\tilde{\xi}_T$ is only applied to those derived from the best hypothesis for a given input. We set the weights as $\gamma_{v,S} = 1$ and $\gamma_{v,\xi_T} = 5$.

B.2.2 Cyclic Consistency Loss \mathcal{L}_V

The cyclic consistency loss, \mathcal{L}_V , for the vision cycle only matches the input image, I , and its rendered reconstruction, \hat{I} , via pixelwise and perceptual metrics:

$$\mathcal{L}_V = \sum_{i=1}^{n_h} p_i \left[\frac{\gamma_{v,I}}{4N_p} \|I - \hat{I}_i\|_1 + \frac{\gamma_{v,p}}{N_b} \sum_{j=1}^{N_b} \frac{\|c_{i,j} - \hat{c}_{i,j}\|_1}{B_j N_{p,j}} \right], \quad (11)$$

where the first term is the pixelwise L_1 loss (normalized by the number of pixels, N_p , and channels) and the second term is the perceptual loss over critic layers (normalized by the channel dimensionality of block j , denoted B_j , and number of spatial features, $N_{p,j}$), both weighted by hypothesis probability p_i . We note that $N_b = 5$ is the number of discriminator block outputs we utilize (one per residual block in C_I), while $c_{i,j}$ and $\hat{c}_{i,j}$ are the feature maps from critic block j and hypothesis i , for I and \hat{I}_i , respectively. This perceptual loss is quite similar to those used in [42, 77]. We set the weights as $\gamma_{v,I} = 40$ and $\gamma_{v,p} = 200$.

B.3. Regularization Losses

The final loss term is composed of an assortment of *regularization objectives*:

$$\mathcal{L}_R = \gamma_{r,m} [L_m(M) + L_m(\widetilde{M})] + \quad (12)$$

$$\gamma_{r,\delta} [L_\delta(\delta) + L_\delta(\tilde{\delta})] + \quad (13)$$

$$L_v(v, \tilde{v}) + L_D(\tilde{\xi}_T, \tilde{R}, \tilde{t}) + \gamma_{r,A} L_A(\tilde{T}_I, \tilde{R}), \quad (14)$$

where L_m is a mesh regularization loss (for preventing pathological geometry), L_δ promotes smoothness in the nodal offset vector (similar in purpose to L_m), L_v regularizes v to be approximately Gaussian (to assist in our *ex-post* density estimation procedure for removing shape-conditioning in the generative process), L_D regularizes the pose hypothesis distributions (as well as the latent texture ones), and L_A denotes

the adversarial texture-pose disentanglement loss (which reduces the dependence of the texture on the pose). We set the weights $\gamma_{r,m} = 0.05$, $\gamma_{r,\delta} = 0.05$, and $\gamma_{r,A} = 0.35$.

We next expand on these terms in detail. Recall that $M = (\mathcal{V}_M, \mathcal{E}, \mathcal{F})$ is a nodal deformation of the template mesh. The mesh regularizer borrows three terms from the literature:

$$L_m(M) = \frac{\gamma_{r,m,e}}{|\mathcal{E}|} \sum_{(v_1, v_2) \in \mathcal{E}} \|v_1 - v_2\|_2^2 + \quad (15)$$

$$\gamma_{r,m,\ell} L_{\text{Lap}}(M) + \gamma_{r,m,f} L_{\text{Flat}}(M), \quad (16)$$

where the first term regularizes the edge lengths [76, 14], the second is a Laplacian loss [45] (designed to prevent excessive curvature), and the third is a flatness loss [45] (which encourages local smoothness of the surface normals). We set the weights $\gamma_{r,m,e} = 1500$, $\gamma_{r,m,\ell} = 10$, and $\gamma_{r,m,f} = 1$.

The next term regularizes the smoothness of the nodal perturbation directly:

$$L_\delta(\delta) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \sum_{d=1}^3 \mathbb{V}[\delta[f]]_d, \quad (17)$$

where $\mathbb{V}[\delta[f]]_d$ is the empirical variance of δ values over triangle face $f \in \mathcal{F}$, for spatial dimension d .

The regularizer on v is simply an SWD loss on the latent shape (similar to the one used for $\tilde{\xi}_T$):

$$L_v(v, \tilde{v}) = \gamma_{r,v} \mathfrak{D}_{\text{SWD}}[v \parallel \eta] + \gamma_{r,\tilde{v}} \mathfrak{D}_{\text{SWD}}[\tilde{v} \parallel \eta], \quad (18)$$

where $\eta \sim \mathcal{N}(0, I)$, and we set $\gamma_{r,v} = 5$ and $\gamma_{r,\tilde{v}} = 0.1$ (because the shape critic C_s regularizes \tilde{v} as well).

We next consider the “diversification” losses on the pose hypotheses, as well as on the resulting texture hypotheses:

$$L_D(\tilde{\xi}_T, \tilde{R}, \tilde{t}) = -\frac{\gamma_{r,d,R}}{\pi} \sum_{i=1}^{n_h} \sum_{j=1}^{n_h} p_i p_j d_R(\tilde{R}_i, \tilde{R}_j) + \quad (19)$$

$$\frac{\gamma_{r,d,\xi_T}}{n_h \dim(\xi_T)} \sum_{k=1}^{n_h} \|\tilde{\xi}_{T,k} - \tilde{\xi}_{T,b}\|_2^2 + \quad (20)$$

$$\frac{\gamma_{r,d,t}}{3n_h} \sum_{k=1}^{n_h} \|\tilde{t}_k - \tilde{t}_b\|_2^2, \quad (21)$$

where the three terms encourage diversity in the rotations \tilde{R} , and discourage it for $\tilde{\xi}_T$ and \tilde{t} . The index b refers to the “best” hypothesis (i.e., $\arg \max_j p_j$). We set the weights $\gamma_{r,d,R} = 0.1$, $\gamma_{r,d,\xi_T} = 1$, and $\gamma_{r,d,t} = 10$. We remark that we are trying to encourage diversity in the rotations (as in [40]) in an effort to spur exploration, but found reducing it for the translations helped with numerical stability (whereas, for the latent shape, we desire the same texture regardless of pose).

The final regularization term is designed to encourage disentanglement of rigid pose and texture. We trained an

adversary network, A_R , which attempted to predict the rotation component \tilde{R} of our inferred Euclidean transform \tilde{E} , from the texture image \tilde{T}_I . Note that \tilde{T}_I depends directly on \tilde{R} through the unprojection. We implemented A_R as a ResNet-20 [20], which regressed a quaternion from the image, followed by conversion to a rotation matrix. We trained A_R with one gradient descent step per generator update (i.e., for the main model), using Adam [37] (learning rate 0.0005 and the same β values as for the main model). Adversary training is done via minimization of $d_R(\tilde{R}, A_R(\tilde{T}_I))$. Hence, our main model’s regularization loss is given by

$$L_A(\tilde{T}_I, \tilde{R}) = -d_R(\tilde{R}_b, A_R(\tilde{T}_{I,b})), \quad (22)$$

where we note that this loss is only applied to the *best* hypothesis (i.e., with highest estimated probability), denoted with index b .

B.4. Critic Details

We next discuss the details of the critic architectures. All critics were optimized with Adam [37] with a learning rate of 0.0001, used the WGAN-GP loss formulation [2, 16] with a “drift” penalty [35] on the mean squared output value, and trained with one descent step per generator update. Gradient and drift penalty weights were set to 10 and 0.01, respectively. An L_2 weight decay of 10^{-3} was also applied to all critics, as in [16]. All critics have a scalar output.

The details per critic model are given as follows:

- The critic on generated images (i.e., sampled renders), $C_I(\tilde{I})$, is given by a standard WGAN-GP convolutional discriminator composed of residual blocks: an initial block without downsampling, followed by four blocks with downsampling, and then a ReLU, mean pooling, and a final linear layer. Hidden channel dimensionalities go from 64 to 1024 in factors of two. We reiterate that the use of a *soft* rasterizer [46] and mesh of limited resolution (with interpolated per-triangle colouring), in addition to a weaker lighting model, leads to a fundamental domain gap compared to real images. Hence, we train the image critic C_I with a mix of real images and inferred image reconstructions: with probability $\rho_r = 0.5$, we replace an image I with its vision cycle (VC) reconstruction \tilde{I} . This is designed to increase distributional overlap, which is known to stabilize adversarial training (e.g., [31]).
- The critic on generated texture images, $C_T(\tilde{T}_I)$, employs the same architecture as that of C_I . However, it is trained with the texture images from the vision cycle (as we do not assume access to ground truth textures).
- The critic on inferred shapes, $C_s(\tilde{v}, \tilde{M})$, uses two networks. The first processes \tilde{M} to a 128D vector \tilde{y}_M ,

and is given by LBA (512) BA. The second is a vector critic on the concatenation (\tilde{v}, \tilde{y}_M) , with architecture LBA (512, 256, 128).

C. Experimental Details

C.1. Training Details

All models were implemented in PyTorch [58], and optimized with Adam [37] ($\beta_1 = 0.5, \beta_2 = 0.99$). The learning rate was fixed to 0.0004, using a batch size of 64. To provide normalized input, we scaled all input shapes S into the unit bounding box, and took 1000 points (with normals) per data shape. When computing losses, we also sampled 1000 points from our template.

C.1.1 Curriculum Learning

To encourage stable training, we follow a learning curriculum for our training regime, inspired by techniques in domain randomization [70, 71]. In the first stage, we train the graphics cycle (GC) mapping from the input shape, P , to the canonically deformed template, M , for n_{T_1} iterations. We then define a “domain randomized” version of the GC (abbreviated DR-GC), where Euclidean pose E and texture T are randomly sampled from hand-crafted latent distributions, rather than via the learned densities defined by ξ_p and ξ_T , respectively. The pose distribution is chosen to simulate a camera sampled from the upper-hemisphere of the object, along with a random displacement in a box around the origin. The texture distribution is formed by choosing a random 3D plane in space and colouring the vertices on each side of the plane a constant, randomly chosen colour. The latent texture reconstruction objective is replaced by a loss between the textures T and \hat{T} instead. We train with this DR-GC for n_{T_2} iterations. We then introduce the VC and train it with the DR-GC for n_{T_3} iterations, followed by introducing the normal GC (with both VC and DR-GC) for n_{T_4} iterations. For this fourth stage, we linearly anneal in the losses from the GC, while annealing out the losses from the DR-GC at the same time. The final stage utilizes the full GC and VC for the remaining n_{T_5} iterations. We set $n_{T_1} = 5000, n_{T_2} = 3000, n_{T_3} = 3000, n_{T_4} = 3000$, and $n_{T_5} = 10^5 - \sum_{i=1}^4 n_{T_i}$.

C.1.2 Hyper-Parameter Details

All hyper-parameters were chosen for visual quality, and the same across all categories, except for planes (see Sec. A and B). For the planes category, we found that the model struggled to generate images of equivalent quality to the other categories, for both the vision and graphics cycles. We speculate this is potentially due to the smaller size of the objects relative to the image, as well as the degenerate cases induced when viewing planes from the side.

We made two major changes to mitigate such issues, specific to the planes category. First, due to the relative conformity of plane shapes, we found that utilizing a specialized template was helpful. We used a plane model from the ShapeNet training set, and computed a UV parametrization via boundary first flattening (with 16 auto-placed cones) [68]. Unlike for other categories, we let the template vertex positions be updated during learning as well. Second, we altered several of the hyper-parameters, as follows: $\gamma_{v,I} = 100, \gamma_{v,p} = 100, \gamma_{r,v} = 0.5, \gamma_{r,a} = 0, \dim(v) = 128, \gamma_{g,v}, \gamma_{g,I} = 0.5, \gamma_{g,R} = 1, \gamma_{g,\xi_T} = 1, \gamma_{g,TR} = 10, \gamma_{g,T} = 1, \gamma_{v,\xi_T} = 1, \gamma_{r,m} = 0.005$, and $\gamma_{r,\delta} = 0.1$. Note that $\gamma_{g,TR}$ is the weight coefficient on a nodal texture reconstruction loss term in the graphics cycle,

$$L_{g,TR} = \frac{\gamma_{g,TR}}{3n_h|\mathcal{V}|} \sum_{i=1}^{n_h} \|T - \hat{T}_i\|_1, \quad (23)$$

which was not used for the other categories. The plane model with these altered hyper-parameters is used for all results and images.

For completeness, we also record the quantitative evaluations of the *unmodified* planes model (i.e., the one having identical hyper-parameters to the other categories): for reconstruction, F-score (τ), F-score (2τ), and Chamfer were 63.2, 75.8, and 0.55 (Table 1); for conditional generation, IS, FID, and KID were 2.69, 65.9, and 6.11 (see Table 2); and for unconditional generation, IS, FID, and KID were 2.65, 67.8, and 6.38 (see Table 2).

For all hyper-parameters, we expect that additional experimentation (including category-specific hyper-parameter search) would further improve results in general.

C.2. Dataset Details

We show the number of data points (images and shapes) in the following table:

Dataset	Car	Plane	Sofa	Chair	Cabinet
Train-S ^{SN}	5997	3236	2539	5423	1258
Train-I	143928	77664	60936	130152	30192
Test-S ^{P2M}	7496	4045	3173	6778	1572
Test-I	36000	19416	15240	32352	7560

The appended “-S” and “-I” refer to 3D shapes and 2D images, respectively. The SN and P2M superscripts refer to shapes drawn from ShapeNet (SN) itself versus the Pixel2Mesh (P2M) 3D models data. Note that these are different because ShapeNet shapes are given in pose-aligned form (canonical object space), whereas the data for Pixel2Mesh uses camera-space shape models (with multiple poses, and thus meshes, per ShapeNet shape).

C.3. 3D Reconstruction

We use the same testing set as [76] for fair comparison (as well as the same evaluation metrics). For a given ground

truth image and associated scene-space shape ($I_{\text{gt}}, P_{\text{gt}}$), we run $f_{I \rightarrow S}(I)$ to obtain our Euclidean transformed inferred shape, \tilde{M}_E . We evaluate reconstruction accuracy by (i) translating and scaling our output mesh to match P_{gt} , (ii) uniformly sampling points from the surface of \tilde{M}_E equal to the number of points in P_{gt} , and (iii) computing Chamfer distance and F-score between our output and the ground truth shape using the Kaolin library [30].

For (i), we first mean-centre the mesh. Then, we approximately correct for the scales induced by the differing camera coordinates. Let d_z and f_ℓ be the distance along the optical axis from the camera (pre-centring) and the focal length of the camera, respectively, for the ground-truth data. Let \hat{d}_z and \hat{f}_ℓ be the same for our output (pre-centring). Then our scaling factor is written

$$s = \frac{\hat{d}_z}{\hat{d}_z} \frac{\hat{f}_\ell}{f_\ell}. \quad (24)$$

We remark here that, due to the weak supervision, our model has weak constraints in terms of scale and distance from the camera; for instance, a slightly smaller shape \tilde{M} , but closer to the camera (via \tilde{t}_z), may still produce a valid solution (in terms of the VC losses). One thus might expect an additional scalar scaling factor is needed to correct for this (e.g., based on the variance of the point set, as often used in Procrustes analyses [3]). However, our results did not utilize one, as it reduced performance compared to just using s alone.

C.4. Generative Modelling

We next describe the generative modelling experiments in greater detail. In all cases, we used the rendered data from [7], and the train-test split from [76]. Any render from a shape in the test set was placed in the test set for the renders; thus, our evaluation (using Inception Score [67], Frechet Inception Distance [24], and Kernel Inception Distance [4]) utilizes only images that were not seen in training. We used 50K samples for metric evaluation.

For our generative adversarial network (GAN) baseline [15], we used a WGAN-GP generator for 64×64 images, as implemented in the Mimicry library [43], trained for 50K iterations. Hyper-parameter tuning was not necessary for strong performance. For our variational autoencoder (VAE) baseline, we used the vanilla convolutional VAE architecture [38, 66] from the PyTorch-VAE library [69]. We tuned the learning rate and chose the β value [25] (the weight on the KL-divergence term) by iteratively decreasing it from the default value until the FID increased. Note that both baselines were modified to handle the additional alpha channel (i.e., the mask) present in the data (and produced by the differentiable renderer in our main model).

For our unconditional generation procedure, based on *ex-post* density fitting, we use a Gaussian Mixture Model with 10 components, fit using the scikit-learn library [59].

We use only the v values inferred from the graphics cycle (i.e., only those coming from real *shapes*, via $v = f_v(P)$).

C.4.1 Ablation Experiments

We also considered the effect of ablating (i) the pose-texture disentanglement loss (i.e., the objective maximizing the error of the adversary A_R) and (ii) the texture image critic (i.e., the term minimizing the error of C_T). We find the ablations lower the generative image quality: for the cars model, in terms of IS, FID, and KID, we obtain (i) 3.07, 131.2, and 11.7, and (ii) 2.84, 160.7, and 14.7, respectively, all of which are worse than their non-ablated counterparts.

D. Additional Results Visualizations

Lastly, we display additional visualizations of example results, in the same format as shown in the main text. In Fig. 9, we show 3D reconstruction examples (i.e., running the *vision cycle* on an input image, to obtain an inferred textured mesh, which “explains” the original input image, and renders thereof). In Fig. 12, we present random conditional generations (i.e., applying the shape-to-image modality translation to shape data samples), as well as real images for comparison. Similarly, in Fig. 13, we show random *unconditional* generations (via sampling through our *ex-post* fitted distribution to the latent shape, as described in the main text). For additional comparisons to the baseline generative models (GAN and VAE), we display random samples in Figs. 15 for both (we also show VAE reconstructions in Fig. 11). In Fig. 14, we visualize latent space interpolations, illustrating (a) the disentanglement between shape, pose, and texture and (b) the smoothly manipulable nature of our latent representation. Lastly, in Fig. 10, we show our unsupervised correspondence visualizations. By marking the positions of fixed template vertices across input images, we see that the network learns to assign implicit meaning to a given node. A given template node can thus act like an unsupervised keypoint, which can identify correspondences across images.

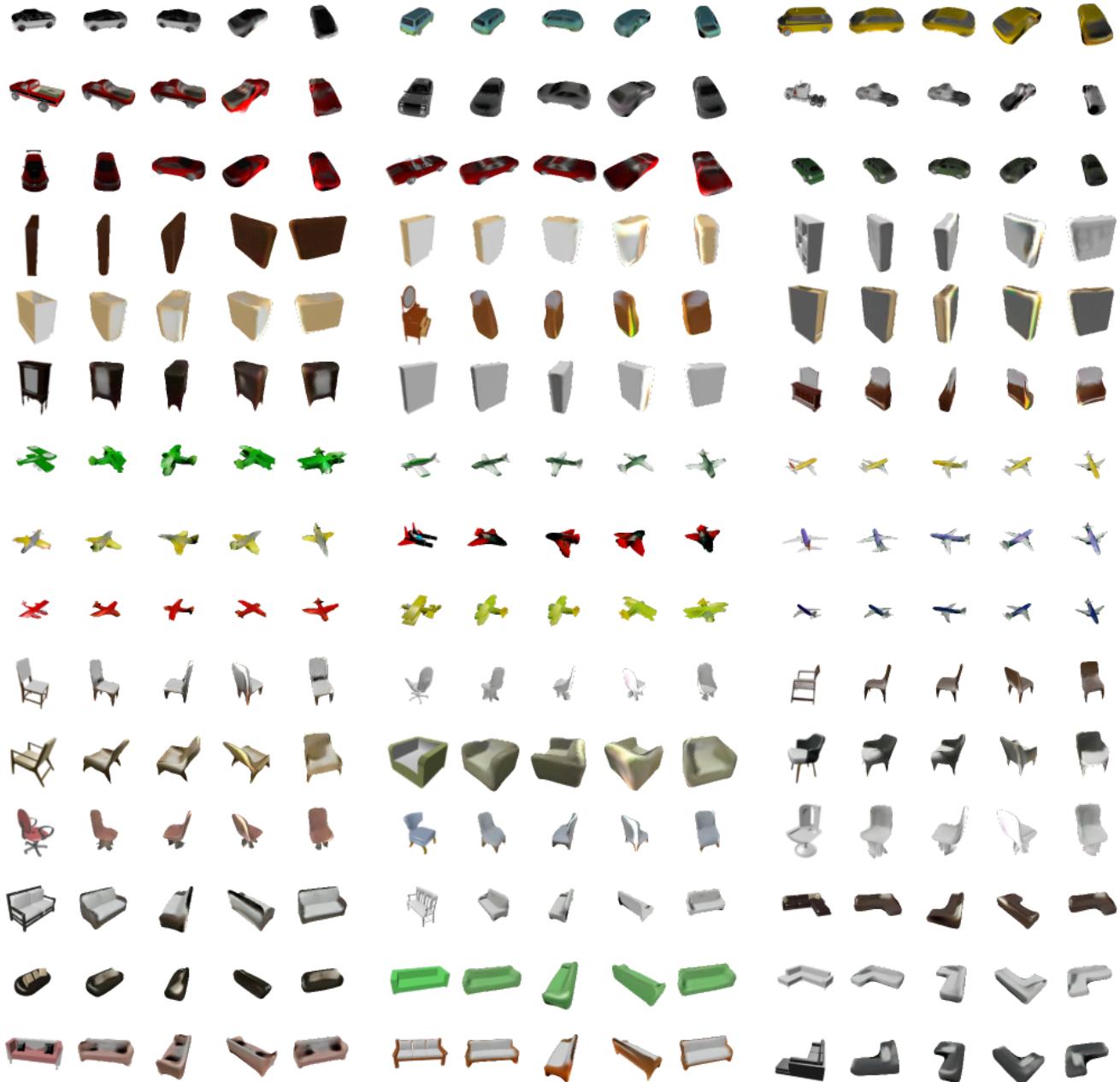


Figure 9. Example 3D inferences. Per inset (left to right): input I , reconstruction \hat{I} , and different fixed views of the inferred textured mesh.



Figure 10. Example unsupervised correspondence visualizations. After inferring the template shape from each given input image I , we project the deformed and transformed template vertices of M_E back into the image I . We highlight the positions of five randomly chosen vertices per image (but choose the same indices for each row). In other words, the chosen indices are the same for a single row (i.e., across columns), but different for every row (i.e., across rows), even for the same category.



Figure 11. Example reconstructions from the VAE baseline model.

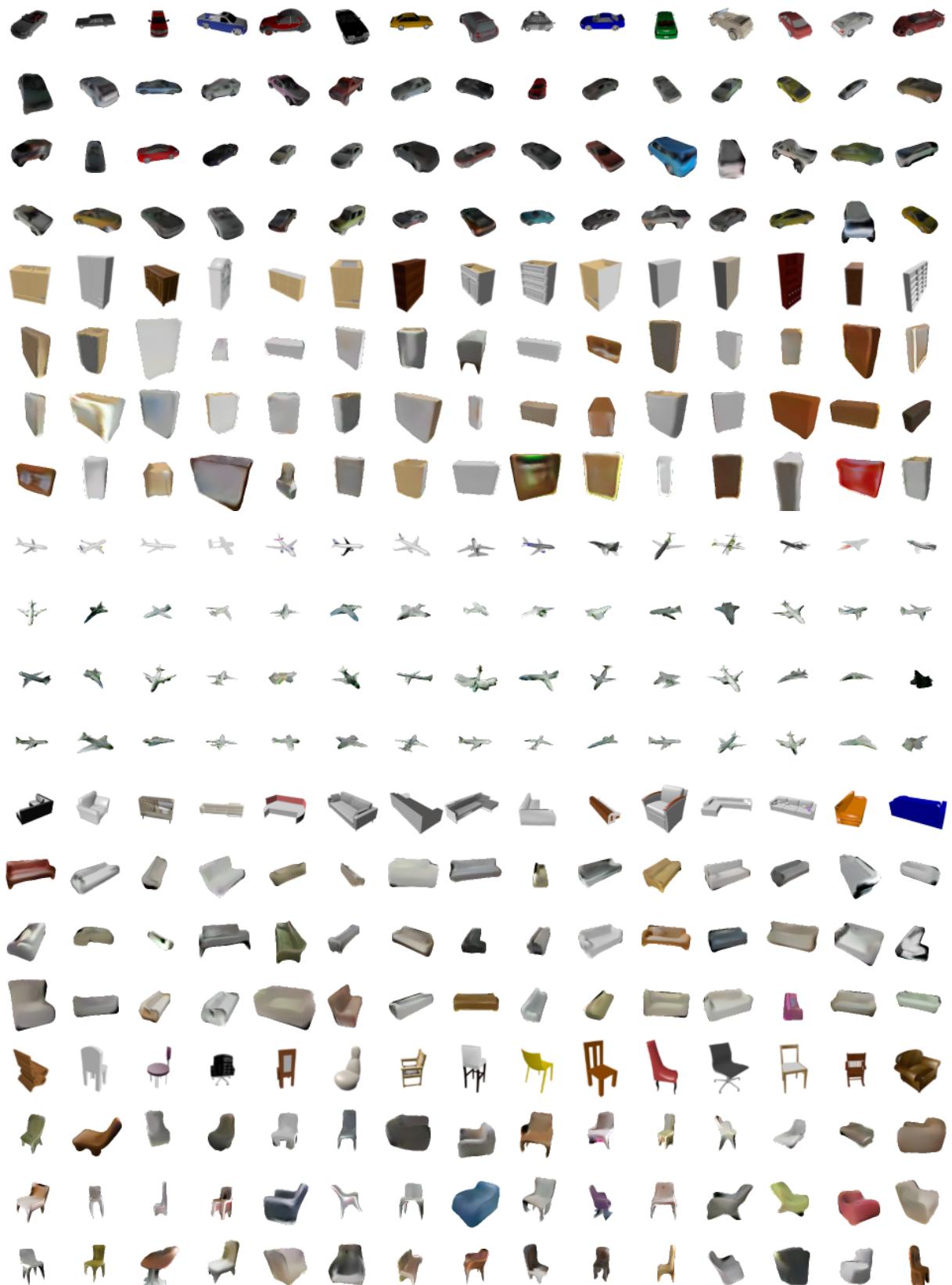


Figure 12. Additional random sampled renders. Per category, we show one row of *real* images, then three rows of generations.



Figure 13. Unconditional generated render samples. As in Fig. 12, for each category, we show one row of real images, then three of samples.

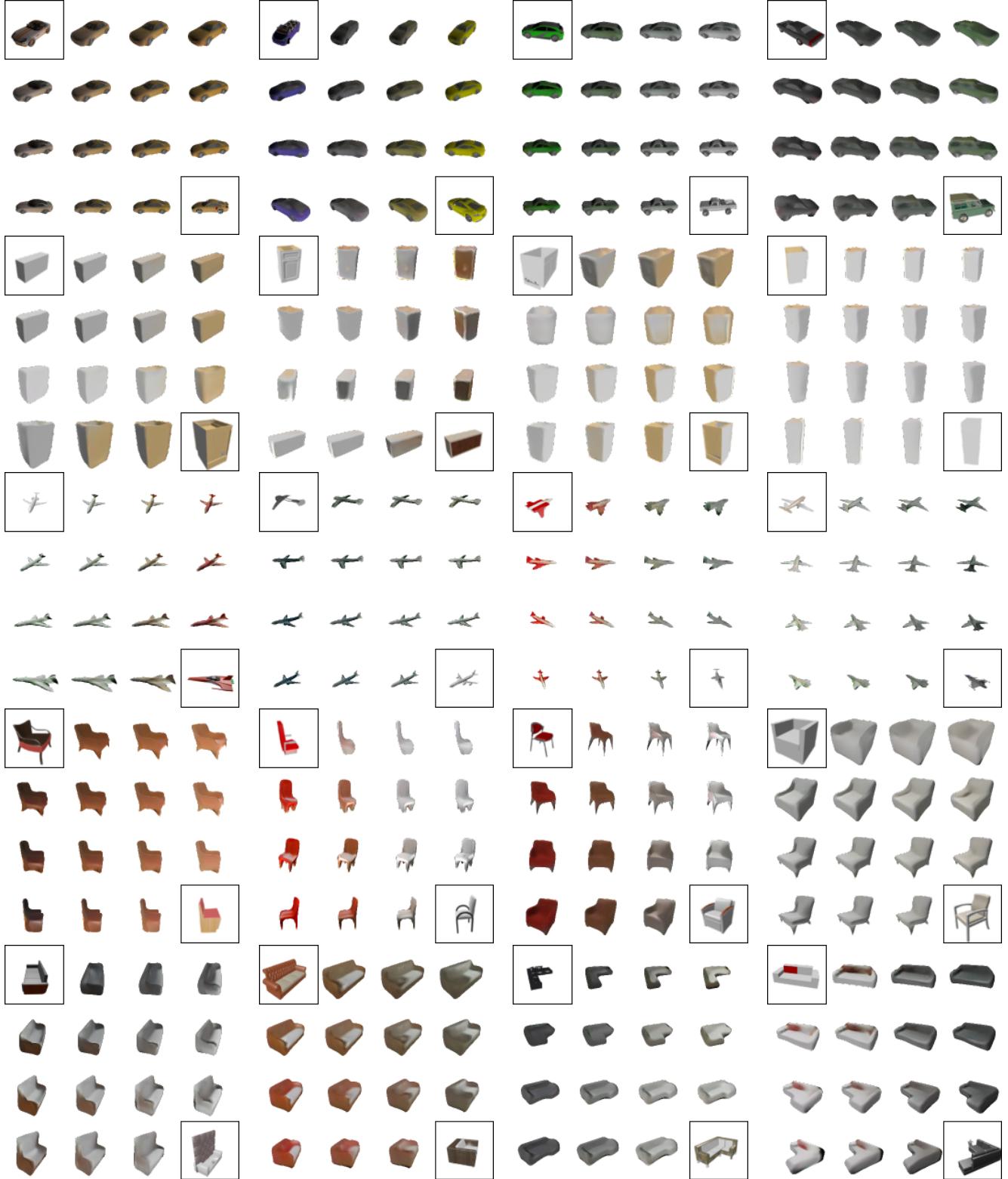


Figure 14. Example 3D latent interpolations. Per-inset: boxed corner images represent start and end points (real images), vertical axis corresponds to interpolation in latent shape (v) and Euclidean pose (E), while the horizontal axis represents interpolation of the latent texture (ξ_T). Non-boxed opposing corners (upper-right and lower-left) may be viewed as *texture transfers*. Note the occasional “flip” of texture (rather than using the Euclidean transform E) to change orientation (e.g., cars, inset four, and planes, inset three).



Figure 15. Example generations from the GAN (rows 1-10) and VAE (rows 11-20) baseline models.