

# Learning Generative Models of Textured 3D Meshes from Real-World Images

Dario Pavllo

Jonas Kohler

Thomas Hofmann

Aurelien Lucchi

Department of Computer Science  
ETH Zurich



Figure 1. **Left:** we focus on GANs, where our generator outputs a *triangle mesh* and a UV-mapped texture. **Middle:** our method learns to synthesize textured 3D meshes given a real-world collection of 2D images. **Top-right:** we showcase a setting where we train a single model to generate all classes. This model successfully disentangles some factors of the 3D environment (e.g. lighting/shadows) without explicit supervision. **Bottom-right:** we also demonstrate a conditional model that generates meshes from 3D semantic layouts.

## Abstract

Recent advances in differentiable rendering have sparked an interest in learning generative models of textured 3D meshes from image collections. These models natively disentangle pose and appearance, enable downstream applications in computer graphics, and improve the ability of generative models to understand the concept of image formation. Although there has been prior work on learning such models from collections of 2D images, these approaches require a delicate pose estimation step that exploits annotated keypoints, thereby restricting their applicability to a few specific datasets. In this work, we propose a GAN framework for generating textured triangle meshes without relying on such annotations. We show that the performance of our approach is on par with prior work that relies on ground-truth keypoints, and more importantly, we demonstrate the generality of our method by setting new baselines on a larger set of categories from ImageNet – for which keypoints are not available – without any class-specific hyperparameter tuning. We release our code at <https://github.com/dariopavlo/textured-3d-gan>

## 1. Introduction

Most of the recent literature in the field of generative models focuses on 2D image generation [36, 54, 22, 3, 23], which largely ignores the fact that real-world images depict 2D projections of 3D objects. Constructing 3D generative models presents multiple advantages, including a fully disentangled control over shape, appearance, pose, as well as an explicit representation of spatial phenomena such as occlusions. While the controllability aspect of 2D generative models can be improved to some extent by disentangling factors of variation during the generation process [53, 40, 21, 22], the assumptions made by these approaches have been shown to be unrealistic without an inductive bias [33]. It is thus unclear whether 2D architectures can reach the same degree of controllability as a native 3D representation.

As a result, a growing line of research investigates learning textured 3D mesh generators in both GAN [39, 4] and variational settings [15]. These approaches are trained with 2D supervision from a collection of 2D images, but require camera poses to be known in advance as learning a joint distribution over shapes, textures, and cameras is particularly difficult. Usually, the required camera poses are estimated from keypoint annotations using a factorization algorithm

such as *structure-from-motion* (SfM) [35]. These keypoint annotations are, however, very expensive to obtain and are usually only available on a few datasets.

In this work, we propose a new approach for learning generative models of textured triangle meshes with minimal data assumptions. Most notably, we do not require keypoint annotations, which are often not available in real-world datasets. Instead, we solely rely on: (i) a single mesh template (optionally, a set of templates) for each image category, which is used to bootstrap the pose estimation process, and (ii) a pretrained semi-supervised object detector, which we modify to infer semantic part segmentations on 2D images. These, in turn, are used to augment the initial mesh templates with a 3D semantic layout that allows us to refine pose estimates and resolve potential ambiguities.

First, we evaluate our approach on benchmark datasets for this task (Pascal3D+ [31] and CUB [45]), for which keypoints are available, and show that our approach is quantitatively on par with the state-of-the-art [39] as demonstrated by FID metrics [16], even though we do not use keypoints. Secondly, we train a 3D generative model on a larger set of categories from ImageNet [6], where we set new baselines without any class-specific hyperparameter tuning. To our knowledge, no prior works have so far succeeded in training textured mesh generators on real-world datasets, as they focus either on synthetic data or on simple datasets where poses/keypoints are available. We also show that we can learn a *single* generator for all classes (as opposed to different models for each class, as done in previous work [39, 4, 15]) and notice the emergence of interesting disentanglement properties (e.g. color, lighting, style), similar to what is observed on large-scale 2D image generators [3].

Finally, we quantitatively evaluate the pose estimation performance of our method under varying assumptions (one or more mesh templates; with or without semantic information), and showcase a proof-of-concept where 3D meshes are generated from sketches of semantic maps (*semantic mesh generation*), following the paradigm of image-to-image translation. In summary, our main contributions are as follows:

- We introduce a new approach to 3D mesh generation that does not require keypoint annotations, enabling its use on a wider range of datasets as well as new image categories.
- We showcase 3D generative models in novel settings, including learning a *single* 3D generator for all categories, and *conditional generation* from semantic mesh layouts. In addition, we provide a preliminary analysis of the disentanglement properties learned by these models.
- We propose a comprehensive 3D pose estimation framework that combines the merits of template-based approaches and semantic-based approaches. We further extend this framework by explicitly resolving pose ambiguities and by adding support to multiple templates.

## 2. Related work

**Differentiable 3D representations.** Recent work in 3D deep learning has focused on a variety of 3D representations. Among *reconstruction* approaches, where the goal is to reconstruct 3D meshes from various input representations, [37] predict signed distance fields from point clouds, [5, 12, 51, 9, 56, 47, 42] predict 3D meshes from images using a voxel representation, and [7] predict point clouds from images. These approaches require some form of 3D supervision, which is only achievable through synthetic datasets. More recent efforts have therefore focused on reconstructing meshes using 2D supervision from multiple views, e.g. [50, 11, 44, 46, 43, 52] in the voxel setting, and [19] using point clouds. However, the multiple-viewpoint assumption is unrealistic on real-world collections of natural images, which has motivated a new class of methods that aim to reconstruct 3D meshes from single-view images. Among recent works, [24, 32, 20, 4, 10, 29] are all based on this setting and adopt a *triangle mesh* representation. Our work also focuses on triangle meshes due to their convenient properties: (i) their widespread use in computer graphics, movies, video games; (ii) their support for UV texture mapping, which decouples shape and color; (iii) the ability of efficiently manipulating and transforming vertices via linear algebra. The use of triangle meshes in deep learning was recently enabled by *differentiable renderers* [34, 24, 32, 4], i.e. renderers that provide gradients w.r.t. scene parameters. Motivated by its support for UV maps, we use *DIB-R* [4] as our renderer of choice throughout this work.

**Keypoint-free pose estimation.** The use of keypoints for pose estimation is limiting due to the lack of publicly available data and an expensive annotation process. Thus, a growing line of research focuses on inferring poses via semi-supervised objectives. To our knowledge, no approach has so far focused on *generation*, but there have been some successful attempts in the *reconstruction* literature. The initial pose estimation step of our framework is most closely related to [10, 29], which both propose approaches for 3D mesh *reconstruction* without keypoints. In terms of assumptions, [10] require a canonical mesh template for each category. Object poses are estimated by fitting the mesh template to the silhouette of the object and by concurrently optimizing multiple camera hypotheses (which helps to deal with the large amount of bad local minima). [29] do not require a mesh template, but instead use object part segmentations from a self-supervised model (*SCOPS* [18]) to infer a 3D semantic template that is matched to the reference segmented image. Based on early experiments, we were unable to individually generalize these methods to *generation* (our goal), which we found to have a lower tolerance to errors due to the intrinsic difficulty in training GANs. Instead, we here successfully combine both ideas (mesh templates *and* semantics) and extend the overall framework with (i) the

optional support for multiple mesh templates, (ii) a principled ambiguity resolution step that leverages part semantics to resolve conflicts among camera hypotheses with similar reprojection errors. We additionally adopt a more general object-part segmentation framework. Namely, we use a pre-trained semi-supervised object detector [17] modified to produce fine-grained semantic templates (Fig. 2), as opposed to SCOPS (used in [29]), which we found to require class-specific hyperparameter tuning.

**Mesh generation.** In the *generation* literature there has been work on voxel representations [48, 9, 41, 49, 55, 2] and point clouds [1, 8]. These approaches require 3D supervision from synthetic data and are thus subject to the same limitations mentioned earlier. To our knowledge, the only approaches that tackle this task on a *triangle mesh* setting using exclusively 2D supervision are [15], which focuses on a VAE setting using face colors (as opposed to full texture mapping) and is thus complementary to our work, and [4, 39], which adopts a GAN setting. In particular, [4] represents the earliest attempt in generating textured 3D meshes using GANs, but their approach cannot supervise textures directly from image pixels. By contrast, the more recent [39] proposes a more comprehensive framework that can model both meshes and UV-mapped textures, which allows for successful application to natural images (albeit with keypoint annotations). We build upon [39], from which we borrow the GAN architecture but substantially rework the supervision strategy to relax the keypoint requirement.

### 3. Method

**Data requirements.** As usual in both the *reconstruction* [24, 20, 10, 29] and *generation* [14, 4, 39] literature, we require a dataset of segmented images. Segmentation masks (a.k.a. *silhouettes*) can easily be obtained through an off-the-shelf model (we use PointRend [26] pretrained on COCO [30]; details in Appendix A.1). Whereas prior approaches require keypoint annotations for every image, we only require an untextured mesh template for each image category, which can be downloaded freely from the web. Optionally, our framework supports multiple mesh templates per category, a choice we explicitly evaluate in sec. 4.2. We note that pose estimation from silhouettes alone can in some cases be ambiguous, and therefore we rely on object part semantics to resolve these ambiguities wherever possible. To this end, we use the semi-supervised, large-vocabulary object detector from [17, 38] to infer part segmentations on all images. We adopt their pretrained model as-is, without further training or fine-tuning, but post-process its output as described in Appendix A.1.

**Dataset preparation.** Since our goal is to apply our method to real-world data that has not been manually cleaned or annotated – unlike the commonly-used datasets CUB [45] and Pascal3D+ [31] – we attempt to automatically detect

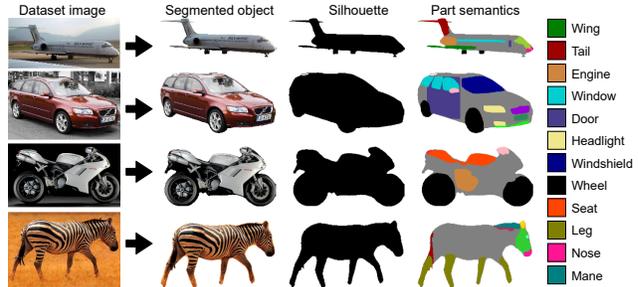


Figure 2. The dataset is initially processed into a clean collection of images with associated object masks and semantic part segmentations. This is done via off-the-shelf models and does not involve any additional data collection. Semantic classes have a precise meaning and are shared between different categories (e.g. wheels appear in both cars and motorbikes).

and remove images that do not satisfy some quality criteria. In particular, objects should not be (i) too small, (ii) truncated, or (iii) occluded by other objects (implementation details in Appendix A.1). This filtering step is tuned for high precision and low recall, as we empirically found that it is beneficial to give more importance to the former. All our experiments and evaluations (sec. 4) are performed on the dataset that results from this step. Finally, sample images and corresponding silhouettes/part segmentations can be seen in Fig. 2, which also highlights how some semantic parts are shared across image categories.

#### 3.1. Pose estimation framework

**Overview.** Most *reconstruction* and *generation* approaches require some form of pose estimation to initialize the learning process. Jointly learning a distribution over camera poses and shapes/textures is extremely challenging and might return a trivial solution that does not entail any 3D reasoning. Therefore, our approach also requires a pose estimation step in order to allow the learning process to converge to meaningful solutions. Our proposed pose estimation pipeline is summarized in Fig. 3: starting from a set of randomly-initialized camera hypotheses for each object instance, we render the mesh template(s) using a differentiable renderer and optimize the camera parameters so that the rendered silhouette matches the target silhouette of the object. At this point, no semantics, colors, or textures are involved, so the approach can lead to naturally ambiguous poses (see Fig. 3 right, for an example). We then introduce a novel ambiguity detection step to select only images whose inferred pose is unambiguous, and use the most confident ones to infer a 3D *semantic template*, effectively augmenting the initial mesh templates with semantic information (more examples of such templates can be seen in Fig. 5). Afterwards, the process is repeated – this time leveraging semantic information – to resolve ambiguities and possibly reinstate images that were previously discarded. The final

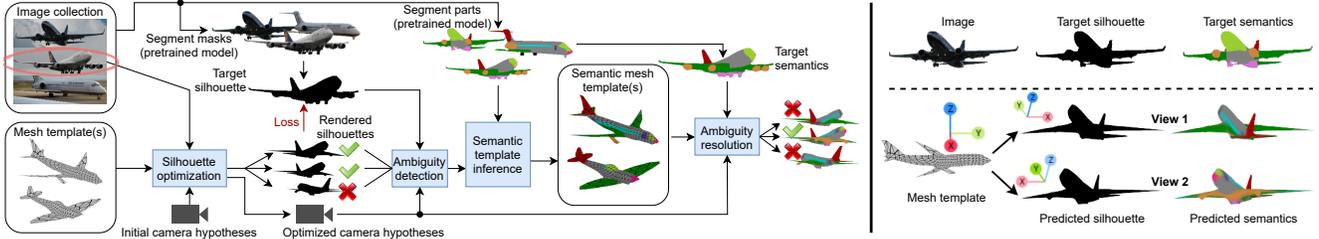


Figure 3. **Left:** schematic overview of the proposed pose estimation pipeline. The left side shows our data requirements (a collection of 2D images and one or more untextured mesh templates). For clarity, we only show the optimization process for the circled airplane, although the *semantic template inference* step involves multiple instances. **Right:** ambiguity arising from opposite poses. The two camera hypotheses produce almost-identical silhouettes which closely approximate the target, but describe opposite viewpoints. This particular example would initially be rejected by our ambiguity detection test, but it would then be resolved once semantics are available.

output is a camera pose for each object as well as a confidence score that can be used to trade off recall (number of available images) for precision (similarity to ground-truth poses). In the following, we describe each step in detail.

**Silhouette optimization.** The first step is a fitting procedure applied separately to each image. Following [10], who observe that optimizing multiple camera hypotheses with differing initializations is necessary to avoid local minima, we initialize a set of  $N_c$  camera hypotheses for each image as described in Appendix A.1. Our camera projection model is the *augmented* weak-perspective model of [39], which comprises a rotation  $\mathbf{q} \in \mathbb{R}^4$  (a unit quaternion), a scale  $s \in \mathbb{R}$ , a screen-space translation  $\mathbf{t} \in \mathbb{R}^2$ , and a perspective correction term  $z_0 \in \mathbb{R}$  which is used to approximate perspective distortion for close objects. We minimize the mean squared error (MSE) in pixel space between the rendered silhouette  $\mathcal{R}(\cdot)$  and the target silhouette  $\mathbf{x}$ :

$$\min_{\mathbf{q}, \mathbf{t}, s, z_0} \|\mathcal{R}(\mathbf{V}_{\text{tpl}}, \mathbf{F}_{\text{tpl}}; \mathbf{q}, \mathbf{t}, s, z_0) - \mathbf{x}\|^2, \quad (1)$$

where  $\mathcal{R}$  is the differentiable rendering operation,  $\mathbf{V}_{\text{tpl}}$  represents the (fixed) mesh template vertices, and  $\mathbf{F}_{\text{tpl}}$  represents the mesh faces. Each camera hypothesis is optimized using a variant of Adam [25] that implements full-matrix preconditioning as opposed to a diagonal one. Given the small number of learnable parameters (8 for each hypothesis), the  $\mathcal{O}(n^3)$  cost of inverting the preconditioning matrix is negligible compared to the convergence speed-up. We provide hyperparameters and more details about this choice in the Appendix A.1. In the settings where we use multiple mesh templates  $N_t$ , we simply replicate each initial camera hypothesis  $N_t$  times so that the total number of hypotheses to optimize is  $N_c \cdot N_t$ . In this case, we compensate for the increase in optimization time by periodically pruning the worst camera hypotheses during optimization. Additionally, in all settings, we start by rendering at a low image resolution and progressively increase the resolution over time, which further speeds up the process. We describe how both strategies are implemented in the Appendix A.1.

**Scoring and ambiguity detection.** All symmetric objects (i.e. many natural and man-made objects) present *ambigu-*

*ous poses*: opposite viewpoints that produce the same silhouette after 2D projection (Fig. 3 right). Similar ambiguities can also arise as a result of noisy segmentation masks, inappropriate mesh templates, or camera hypotheses that converge to bad local minima. Since wrong pose estimates have a significant negative impact on the rest of the pipeline, this motivates the design of an *ambiguity detection* step. Ideally, we would like to accept pose estimates that are both *confident* – using the *intersection-over-union* (IoU) between the rendered/target silhouettes as a proxy measure – and *unambiguous*, i.e. no two camera hypotheses with high IoU should describe significantly different poses. We formalize this as follows: we first score each hypothesis  $k$  as  $(\mathbf{v}_{\text{conf}})_k = (\text{softmax}(\mathbf{v}_{\text{IoU}} / \tau))_k$ , where  $\tau = 0.01$  is a temperature coefficient that gives similar weights to IoU values that are close to the maximum, and low weights to IoU values that are significantly lower than the maximum. Next, we require that highly-confident poses (as measured by  $\mathbf{v}_{\text{conf}}$ ) should describe similar rotations. We therefore construct a pairwise distance matrix  $\mathbf{D}$  of shape  $N_c \times N_c$ , where each entry  $d_{ij}$  describes the geodesic distance between the rotation of the  $i$ -th hypothesis and the rotation of the  $j$ -th hypothesis. Entries are then weighted by  $\mathbf{v}_{\text{conf}}$  across both rows and columns, and are finally summed up, yielding a scalar agreement score  $v_{\text{agr}}$  for each image:

$$\mathbf{D} = 1 - (\mathbf{Q}^T \mathbf{Q})^{\circ 2}, \quad v_{\text{agr}} = \|\mathbf{D} \odot (\mathbf{v}_{\text{conf}} \mathbf{v}_{\text{conf}}^T)\|_1 \quad (2)$$

where  $\mathbf{Q}$  is a  $4 \times N_c$  matrix of unit quaternions (one per hypothesis),  $\mathbf{M}^{\circ 2}$  denotes the element-wise square, and  $\odot$  denotes the element-wise product.

The agreement score  $v_{\text{agr}}$  can be roughly interpreted as follows: a score of 0 (best) implies that all *confident* camera hypotheses describe the same rotation (they agree with each other). A score of 0.5 describes two poses that are rotated by 180 degrees from one another<sup>1</sup>. Empirically, we established that images with  $v_{\text{agr}} > 0.3$  should be rejected.

**Semantic template inference.** Simply discarding ambiguous images might significantly reduce the size and diver-

<sup>1</sup>For example, consider a  $\mathbf{D}$  matrix of size  $2 \times 2$ , where entries along the main diagonal are 0, and 1 elsewhere.

sity of the training set. Instead, we propose to resolve the ambiguous cases. While this is hardly possible when we only have access to silhouettes, it becomes almost trivial once *semantics* are available (Fig. 3 right). A similar idea was proposed in [29], who infer a 3D semantic template by averaging instances that are close to a predetermined exemplar (usually an object observed from the left or right side). Yet, our formulation does not require an exemplar but directly leverages samples that have passed the ambiguity detection test. Since our data requirements assume that mesh templates are untextured, our first step in this regard aims at augmenting each mesh template with part semantics. Among images that have passed the ambiguity test ( $v_{agr} < 0.3$ ), we select the camera hypothesis with the highest IoU. For each mesh template, the semantic template is computed using the top  $N_{top} = 100$  images assigned to that template, as measured by the IoU. Then, we frame this step as an optimization problem where the goal is to learn vertex colors while keeping the camera poses fixed, minimizing the MSE between the rendered (colored) mesh template and the 2D image semantics, averaged among the top samples:

$$\min_{\mathbf{C}_{tpl}} \frac{1}{N_{top}} \sum_i \|\mathcal{R}(\mathbf{V}_{tpl}, \mathbf{F}_{tpl}, \mathbf{C}_{tpl}; \mathbf{q}_i, \mathbf{t}_i, s_i, z_{0i}) - \mathbf{C}_i\|^2, \quad (3)$$

where  $\mathbf{C}_{tpl}$  represents the vertex colors of the template and  $\mathbf{C}_i$  denotes the 2D semantic image. For convenience, we represent  $\mathbf{C}_{tpl}$  as a  $K \times N_v$  matrix, where  $N_v$  is the number of vertices and  $K$  is the number of semantic classes (color channels, not necessarily limited to 3), and  $\mathbf{C}_i$  is a  $K \times N_{pix}$  matrix, where  $N_{pix}$  is the number of image pixels. In the Appendix A.1, we derive an efficient closed-form solution that requires only a single pass through the dataset. Examples of the resulting semantic templates are shown in Fig. 5.

**Ambiguity resolution.** In the last step of our pose estimation pipeline, we repeat the scoring process described in “Scoring and ambiguity detection” with the purpose of resolving ambiguities. Instead of evaluating the scores on the IoU, however, we use the mean intersection-over-union (mIoU) averaged across semantic classes. Since our inferred semantic templates are continuous, we adopt a smooth generalization of the mIoU (weighted Jaccard similarity) in place of the discrete version:

$$mIoU = \frac{1}{K} \sum_k \frac{\|\min(\hat{\mathbf{C}}_k, \mathbf{C}_k)\|_1}{\|\max(\hat{\mathbf{C}}_k, \mathbf{C}_k)\|_1} \quad (4)$$

where  $\hat{\mathbf{C}}_k$  is the rendered semantic class  $k$  and  $\min, \max$  (performed element-wise) represent the weighted intersection and union, respectively. We then recompute the confidence scores and agreement scores as before (using the mIoU as a target metric), discard the worst 10% images in terms of mIoU as well as those whose  $v_{agr} > 0.3$ , and select the best hypothesis for each image as measured by the mIoU. We found no practical advantage in repeating the semantic template inference another time, nor in re-

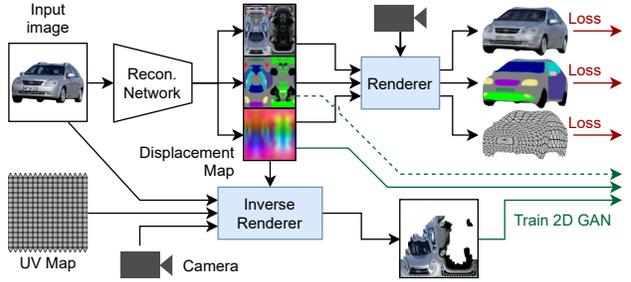


Figure 4. Generation framework using the *convolutional mesh* representation. Images are fed into a network trained to reconstruct meshes (parameterized as 2D displacement maps), given camera poses. The meshes are then used to project natural images onto the UV map. Finally, the resulting partial textures, displacement maps, and (optionally) predicted semantics are used to train a 2D convolutional GAN in UV space.

optimizing/fine-tuning the camera poses using semantics. We show this quantitatively in sec. 4.2 and discuss further details on various exploratory attempts in Appendix A.4.

### 3.2. Generation framework

The camera poses obtained using the approach described in sec. 3.1 can be used to train a generative model as shown in Fig. 4. For this component, we build upon [39], from which we borrow the *convolutional mesh* representation and the GAN architecture. Our generation approach mainly consists of three steps. (i) Given a collection of images, segmentation masks, and their poses<sup>2</sup>, we train a reconstruction model to predict mesh, texture, and semantics given only the 2D image as input. Although predicted textures are not used in subsequent steps (the GAN learns directly from image pixels), [39] observe that predicting textures during training has a beneficial regularizing effect on the mesh, and therefore we also keep this reconstruction term. Unlike [39] (where semantics were not available), however, we also predict a 3D semantic part segmentation in UV space, which provides further regularization and enables interesting conditional generation settings (we showcase this in sec. 4.2). As in [39], we parameterize the mesh as a 2D displacement map that deforms a sphere template in its tangent space. (ii) Through an inverse rendering approach, image pixels are projected onto the UV map of the mesh, yielding partially-occluded textures. Occlusions are represented as a binary mask in UV space. (iii) Finally, displacement maps and textures are modeled in UV space using a standard 2D convolutional GAN, whose training strategy compensates for occlusions by masking inputs to the discriminator.

**Architecture.** Our experiments (sec. 4) analyze two different settings: **A** where we train a separate model for each category, and **B** where we train a single model for all cat-

<sup>2</sup>In [39], poses are estimated via structure-from-motion on ground-truth keypoints. In this work, we use our proposed approach (sec. 3.1).

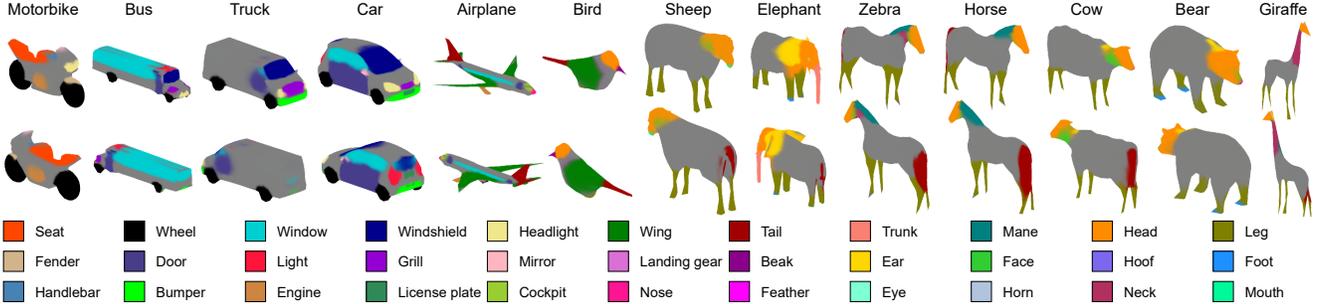


Figure 5. Learned 3D semantic templates. We show one template per category from two views (front/back). Colors are exaggerated for presentation purposes, but in practice the probability maps are smoother. We also highlight how semantic parts are shared among categories.

egories. In setting **A**, we reuse similar reconstruction and GAN architectures to [39] in order to establish a fair comparison with their approach. We only modify the output head of the reconstruction model, where we add  $K$  extra output channels for the semantic class prediction ( $K$  depends on the category). In setting **B**, we condition the model on the object category by modifying all BatchNorm layers and learning different gain and bias parameters for each category. Additionally, in the output head we share semantic classes among categories (for instance there is a unique output channel for *wheel* that is shared for buses, trucks, etc.; see Fig. 5). We do not make any other change that would affect the model’s capacity. As for the GAN, in both **A** and **B**, we use the same architecture as [39]. Further details regarding hyperparameters, implementation and optimizations to improve rendering speed can be found in Appendix A.1.

**Loss.** The reconstruction model is trained to jointly minimize the MSE between (i) rendered and target silhouettes, (ii) predicted RGB texture and target 2D image, (iii) predicted semantic texture (with  $K$  channels) and target 2D semantic image. As in [39], we add a smoothness loss to encourage neighboring faces to have similar normals. Finally, the availability of mesh templates allows us to incorporate a strong shape prior into the model via a loss term that can be regarded as an extreme form of semi-supervision: on images with very confident poses (high IoU), we provide supervision directly on the predicted 3D vertices by adding a MSE loss between the latter and the vertices of the mesh template (i.e. our surrogate ground-truth), only on the top 10% of images as measured by the IoU. This speeds up convergence and helps with modeling fine details such as wings of airplanes, where silhouettes alone provide a weak learning signal from certain views. This step requires *remeshing* the templates to align them to a common topology, which we describe in Appendix A.1.

## 4. Experiments

We quantitatively evaluate the aspects that are most central to our approach: pose estimation and generation quality.

**Pose estimation.** On datasets where annotated keypoints are available, we compare the poses estimated by our

approach to poses estimated from *structure-from-motion* (SfM) on ground-truth keypoints. Since the robustness of SfM depends on the number of visible keypoints, we never refer to SfM poses as “ground-truth poses”, as these are not available in the real-world datasets we use. Nonetheless, we believe that SfM poses serve as a good approximation of ground-truth poses on most images. Our evaluation metrics comprise (i) the *geodesic distance* (GD) between the rotation  $\mathbf{q}$  predicted by our approach and the SfM rotation  $\mathbf{p}$ , defined as  $GD = 1 - (\mathbf{p} \cdot \mathbf{q})^2$  for quaternions, where  $GD \in [0, 1]$ <sup>3</sup>; and (ii) the *recall*, which measures the fraction of usable images that have passed the ambiguity detection test. We evaluate pose estimation at different stages: after silhouette optimization (where no semantics are involved), and after the semantic template inference. Additionally, we compare settings where only one mesh template per category is available, and where multiple mesh templates are employed (we use 2–4 templates per category).

**Generative modeling.** Following prior work on textured 3D mesh generation with GANs [39], we evaluate the Fréchet Inception Distance (FID) [16] on meshes rendered from random viewpoints. For consistency, our implementation of this metric follows that of [39]. Since our pose estimation framework discards ambiguous images and the FID is sensitive to the number of evaluated images, we *always* use the full dataset for computing reference statistics. As such, there is an incentive for optimizing both *GD* and *recall* metrics as opposed to trading one off for the other. Finally, consistently with [39], we generate displacement maps at  $32 \times 32$  resolution, textures at  $512 \times 512$ , and sample from the generator using a truncated Gaussian at  $\sigma = 1.0$ .

### 4.1. Datasets

We evaluate our approach on three datasets: CUB-200-2011 (CUB) [45], Pascal3D+ (P3D) [31], and a variety of classes from ImageNet [6]. The first two provide keypoint annotations and serve as a comparison to previous work, whereas on the latter we set new baselines. Combining all

<sup>3</sup>More commonly known as *cosine distance* when quaternions are used to describe orientations, as in our case.

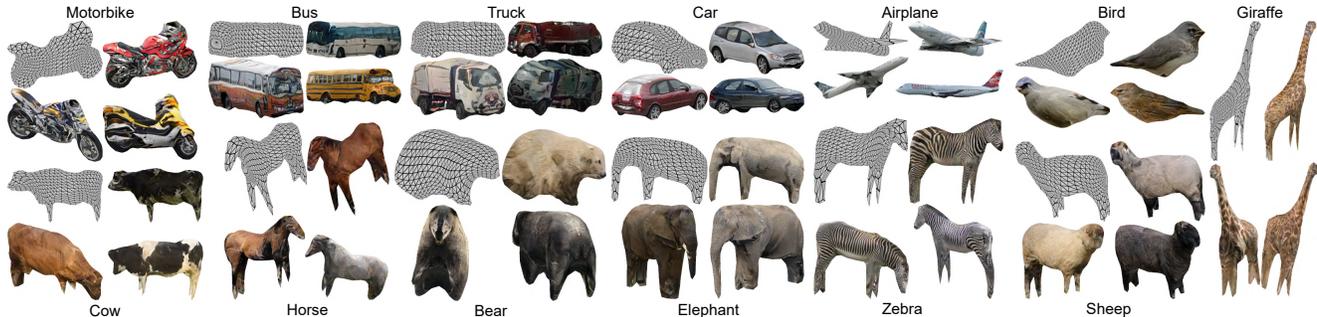


Figure 6. Qualitative results for all 13 classes used in our work. For each class, we show one wireframe mesh on the left, the corresponding textured mesh on the right, and two additional textured meshes on the second row. Meshes are rendered from random viewpoints.

Setting	Step	Bird		Car		Airplane	
		GD(1)	GD (Recall)	GD(1)	GD (Recall)	GD(1)	GD (Recall)
Single template	Silhouette	0.47	0.35 (52%)	0.12	<u>0.05</u> (75%)	0.31	0.28 (85%)
	Semantics	<b>0.29</b>	<b>0.24</b> (74%)	0.11	0.06 (84%)	0.25	0.18 (78%)
	Repeat x2	<b>0.29</b>	<b>0.24</b> (76%)	0.15	0.11 (85%)	0.24	0.17 (75%)
Multiple templates	Silhouette	0.47	0.33 (44%)	0.10	<u>0.05</u> (78%)	0.28	0.22 (81%)
	Semantics	<u>0.32</u>	<u>0.27</u> (76%)	<b>0.06</b>	<b>0.04</b> (88%)	<u>0.22</u>	<b>0.15</b> (79%)
	Repeat x2	<u>0.32</u>	<u>0.27</u> (78%)	<u>0.07</u>	<u>0.05</u> (89%)	<b>0.21</b>	0.16 (80%)

Table 1. Pose estimation results under different settings. Best in **bold**; second best underlined. We report *geodesic distance* (GD; lower = better) after each step and associated recall (higher = better) arising from ambiguity detection. For comparison, we also report GD w/o ambiguity detection, GD(1), assuming 100% recall.

datasets, we evaluate our approach on 13 categories.

**CUB (Birds).** For consistency with prior work, we adopt the split of [39, 20] ( $\approx 6k$  training images). As we work in the unconditional setting, we do not use class labels.

**Pascal3D+ (P3D).** Again, we adopt the split of [39, 20], and test our approach on both *car* and *airplane* categories. Since [39] has only tested on cars, we train the model of [39] on airplanes and provide a comparison here. P3D comprises a subset of images from ImageNet and [39] evaluates only on this subset; for consistency, we adopt the same strategy.

**ImageNet.** Our final selection of classes comprises the vehicles and animals that can be seen in Fig. 5/6. The list of *synsets* used in each class as well as summary statistics are provided in the Appendix A.3. The set of ImageNet classes includes *car* and *airplane*, which partially overlap with P3D. Therefore, when we mention these two classes, we always specify the subset we refer to (ImageNet or P3D). We also note that the dataset is heavily imbalanced, ranging from  $\approx 300$  usable images for *giraffe* to thousands of images for *car*. For this reason, in setting **B** we take measures to balance the dataset during training (Appendix A.1).

## 4.2. Results

**Pose estimation.** We evaluate our pose estimation framework on *bird*, *car*, and *airplane*, for which we have keypoint annotations. Reference poses are obtained using the SfM implementation of [20]. For birds (CUB), the scores are computed on all images, whereas for cars/airplanes they are computed on the overlapping images between P3D and

our ImageNet subset. Results are summarized in Table 1. Interestingly, using multiple mesh templates does not seem to yield substantially different results, suggesting that our approach can work effectively with as little as one template per class. Moreover, incorporating semantic information improves both GD and recall. Finally, we repeat the ambiguity detection and semantic template inference steps a second time, but observe no improvement. Therefore, in our following experiments we only perform these steps once. We further discuss these results in Appendix A.2, where we aim to understand the most common failure modes by analyzing the full distribution of rotation errors. Qualitatively, the inferred 3D semantic templates can be found in Fig. 5.

**Generative model.** We report the FID on ImageNet in Table 2, left (*bird* refers to CUB), where we set new baselines. As before, we compare settings where we adopt a single mesh template *vs* multiple templates. We also showcase a conditional model that learns to synthesize all categories using a single generator (setting **B**). Although this model has the same capacity as the individual models (but was trained to generate all classes at once), we note that its scores are in line with those of setting **A**, and in some classes (e.g.

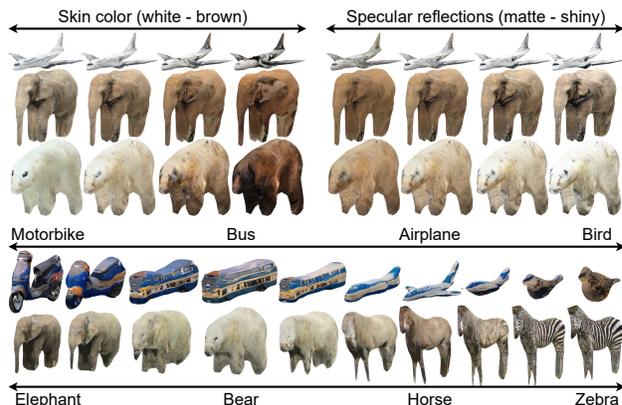


Figure 7. Disentanglement and interpolation in the model trained to generate all classes (setting **B**). **Top:** directions in latent space that correlate with certain style factors, such as skin color and lighting. The effect is consistent across different classes. **Bottom:** interpolation between different classes with a fixed latent code.

Setting	MBike	Bus	Truck	Car	Airplane	Bird	Sheep	Elephant	Zebra	Horse	Cow	Bear	Giraffe	All	Method	Bird (CUB)	Car (P3D)	Airplane (P3D)
Single TPL (A)	107.4	219.3	<b>164.1</b>	<b>30.73</b>	77.84	<b>55.75</b>	173.7	<b>114.5</b>	28.19	113.3	137.0	187.1	157.7	–	Keypoints+SfM [39]	<b>41.56</b>	43.09	147.8*
Multi TPL (A)	107.0	<b>160.7</b>	206.1	32.19	102.2	56.54	<b>155.1</b>	135.9	<b>22.10</b>	107.1	133.0	195.5	<b>126.0</b>	–	Silhouette (single TPL)	73.67	38.16	100.5
Single TPL (B)	94.74	204.98	179.3	39.68	<b>46.46</b>	88.47	169.9	127.6	<u>24.47</u>	<b>106.9</b>	139.4	<b>156.4</b>	176.8	<b>60.82</b>	Silhouette (multi TPL)	88.39	<b>36.17</b>	96.28
Multi TPL (B)	<b>94.03</b>	<u>187.75</u>	204.7	46.11	<u>77.27</u>	77.23	163.8	146.2	31.70	113.4	<b>117.5</b>	189.9	158.0	63.00	Semantics (single TPL)	<u>55.75</u>	<u>36.52</u>	<b>81.28</b>
															Semantics (multi TPL)	56.54	37.56	<u>88.85</u>

Table 2. **Left:** FID of our approach on ImageNet (except *bird*, which refers to CUB). We report results for models trained separately on different classes (setting **A**) and a single model that generates all classes (setting **B**). **Right:** comparison of our FID w.r.t. prior work, using either silhouettes alone or our full pipeline. \* = trained by us; TPL = mesh template(s); lower = better, best in **bold**, second best underlined.

*airplane*) they are significantly better, most likely due to a beneficial regularizing effect. However, we also note that there is no clear winner on all categories. To our knowledge, no prior work has trained a single 3D generator on multiple categories without some form of supervision from synthetic data. Therefore, in one of the following paragraphs we analyze this model from a disentanglement perspective. Next, in Table 2 (right), we compare our results to the state-of-the-art [39] on the *bird*, *car*, and *airplane* categories from CUB/P3D. We find that our approach outperforms [39] on *car* and *airplane* (P3D) – even though we do not exploit ground-truth keypoints – and performs slightly worse on *bird* (CUB). We speculate this is mainly due to the fact that, on CUB, all keypoints are annotated (including occluded ones), whereas P3D only comprises annotations for visible keypoints, potentially reducing the effectiveness of SfM as a pose estimation method. Finally, we point out that although there is a large variability among the scores across classes, comparing FIDs only makes sense within the same class, since the metric is affected by the number of images.

**Qualitative results.** In addition to those presented in Fig. 1, we show further qualitative results in Fig. 6. For animals, we observe that generated textures are generally accurate (e.g. the high-frequency details of zebra stripes are modeled correctly), with occasional failures to model facial details. With regards to shape, legs are sporadically merged but also appear correct on many examples. We believe these issues are mostly due to a pose misalignment, as animals are deformable but our mesh templates are rigid. As part of future work, we would like to add support for articulated mesh templates [28] to our method. As for vehicles, the generated shapes are overall faithful to what one would expect, especially on airplanes where modeling wings is very challenging. We also note, however, that the textures of rare classes (*truck* above all) present some incoherent details. Since we generally observe that the categories with more data are also those with the best results, these issues could in principle be mitigated by adding more images. Finally, we show additional qualitative results in the Appendix A.2.

**Disentanglement and interpolation.** We attempt to interpret the latent space of the model trained to synthesize all classes (setting **B**), following [13]. We identify some directions in the latent space that correlate with characteristics of the 3D scene, including light intensity (Fig. 1, top-right), specular reflections and color (Fig. 7). Importantly, these factors seem to be shared across different classes and are

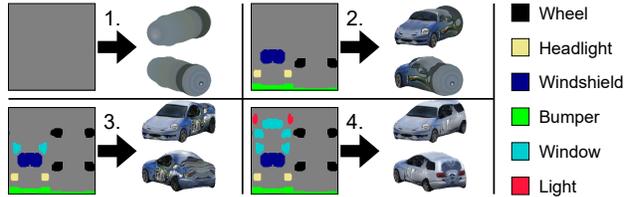


Figure 8. Conditional mesh generation from *semantic layouts*. In this demo, we progressively build a car by sketching its parts, proposing an interesting way of controlling the generation process.

learned without explicit supervision. Although our analysis is preliminary, our findings suggest that 3D GANs disentangle high-level features in an interpretable fashion, similar to what is observed in 2D GANs to some extent (e.g. on pose and style). However, since 3D representations already disentangle appearance and pose, the focus of the disentangled features is on other aspects such as texture and lighting. Fig. 7 (bottom) illustrates interpolation between different classes while keeping the latent code fixed. Style is preserved and there are no observable artifacts, suggesting that the latent space is structured.

**Semantic mesh generation.** Since our framework predicts a 3D semantic layout for each image, we can condition the generator on such a representation. In Fig. 8, we propose a proof-of-concept where we train a conditional model on the *car* class that takes as input a semantic layout in UV space and produces a textured mesh. Such a setting can be used to manipulate fine details (e.g. the shape of the headlights) or the placement of semantic parts.

## 5. Conclusion

We proposed a framework for learning generative models of textured 3D meshes. In contrast to prior work, our approach does not require keypoint annotations, enabling its use on real-world datasets. We demonstrated that our method matches the results of prior works that use ground-truth keypoints, without having to rely on such information. Furthermore, we set new baselines on a subset of categories from ImageNet [6], where keypoints are *not* available. We believe there are still many directions of interest to pursue as future work. In addition to further analyzing disentanglement and exploring more intuitive semantic generation techniques, it would be interesting to experiment with articulated meshes and work with more data.

**Acknowledgments.** This work was partly supported by the Swiss National Science Foundation (SNF), grant #176004.

## References

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning*, pages 40–49, 2018. 3
- [2] Elena Balashova, Vivek Singh, Jiangping Wang, Brian Teixeira, Terrence Chen, and Thomas Funkhouser. Structure-aware shape synthesis. In *2018 International Conference on 3D Vision (3DV)*, pages 140–149. IEEE, 2018. 3
- [3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2
- [4] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Neural Information Processing Systems*, pages 9605–9616, 2019. 1, 2, 3, 16
- [5] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 2
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009. 2, 6, 8
- [7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 605–613, 2017. 2
- [8] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018. 3
- [9] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016. 2, 3
- [10] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoint without keypoints. In *European Conference on Computer Vision*, pages 88–104. Springer, 2020. 2, 3, 4, 14
- [11] JunYoung Gwak, Christopher B Choy, Manmohan Chandraker, Animesh Garg, and Silvio Savarese. Weakly supervised 3d reconstruction with adversarial constraint. In *2017 International Conference on 3D Vision (3DV)*, pages 263–272. IEEE, 2017. 2
- [12] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017. 2
- [13] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9841–9850, 2020. 8
- [14] Paul Henderson and Vittorio Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, pages 1–20, 2019. 3
- [15] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. Leveraging 2d data to learn textured 3d mesh generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 3
- [16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Neural Information Processing Systems*, pages 6626–6637, 2017. 2, 6
- [17] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to segment every thing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4233–4241, 2018. 3, 12
- [18] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 869–878, 2019. 2, 12
- [19] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, pages 2802–2812, 2018. 2
- [20] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 7, 14
- [21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 1
- [22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 1
- [23] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 1
- [24] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916, 2018. 2, 3
- [25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014. 4, 13, 14
- [26] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9799–9808, 2020. 3, 12

- [27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 123(1):32–73, 2017. [12](#)
- [28] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 452–461, 2020. [8](#), [12](#)
- [29] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *European Conference on Computer Vision*, pages 677–693. Springer, 2020. [2](#), [3](#), [5](#), [14](#)
- [30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. [3](#), [12](#)
- [31] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. *International Conference on Learning Representations*, 2019. [2](#), [3](#), [6](#)
- [32] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7708–7717, 2019. [2](#), [16](#)
- [33] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning (ICML)*, 2019. [1](#)
- [34] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision (ECCV)*, pages 154–169. Springer, 2014. [2](#)
- [35] Manuel Marques and João Costeira. Estimating 3d shape from degenerate sequences with missing data. *Computer Vision and Image Understanding*, 113(2):261–272, 2009. [2](#)
- [36] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations (ICLR)*, 2018. [1](#)
- [37] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. [2](#)
- [38] Dario Pavllo, Aurelien Lucchi, and Thomas Hofmann. Controlling style and semantics in weakly-supervised image generation. In *European Conference on Computer Vision (ECCV)*, 2020. [3](#), [12](#)
- [39] Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurélien Lucchi. Convolutional generation of textured 3d meshes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#), [8](#), [14](#)
- [40] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. FineGAN: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. [1](#)
- [41] Edward J Smith and David Meger. Improved adversarial systems for 3d object generation and reconstruction. In *Conference on Robot Learning*, pages 87–96, 2017. [3](#)
- [42] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017. [2](#)
- [43] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2897–2905, 2018. [2](#)
- [44] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2634, 2017. [2](#)
- [45] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. [2](#), [3](#), [6](#)
- [46] Olivia Wiles and Andrew Zisserman. Silnet : Single- and multi-view reconstruction by learning from silhouettes. In Gabriel Brostow Tae-Kyun Kim, Stefanos Zafeiriou and Krystian Mikolajczyk, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 99.1–99.13. BMVA Press, September 2017. [2](#)
- [47] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5d sketches. In *Neural Information Processing Systems*, pages 540–550, 2017. [2](#)
- [48] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016. [3](#)
- [49] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8629–8638, 2018. [3](#)
- [50] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016. [2](#)
- [51] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3d object reconstruction from a single depth view with adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 679–688, 2017. [2](#)

- [52] Guandao Yang, Yin Cui, Serge Belongie, and Bharath Hariharan. Learning single-view 3d reconstruction with limited pose supervision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 86–101, 2018. 2
- [53] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. LR-GAN: layered recursive generative adversarial networks for image generation. In *International Conference on Learning Representations (ICLR)*, 2017. 1
- [54] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2019. 1
- [55] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: Image generation with disentangled 3d representations. In *Neural Information Processing Systems*, pages 118–129, 2018. 3
- [56] Rui Zhu, Hamed Kiani Galoogahi, Chaoyang Wang, and Simon Lucey. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 57–65, 2017. 2

## A. Supplementary material

### A.1. Implementation details

**Dataset preparation.** We infer object silhouettes using PointRend [26] with an X101-FPN backbone, using their pretrained model on COCO [30]. We set the object detection threshold to 0.9 to select only confident objects. As mentioned in sec. 3, we discard object instances that are either (i) too small (mask area  $< 96^2$  pixels), (ii) touch the borders of the image (indicator of possible truncation), or (iii) collide with other detected objects (indicator of potential occlusion). For the object part segmentations, we use the semi-supervised object detector from [17], which can segment all 3000 classes available in Visual Genome (VG) [27] while being supervised only on mask annotations from COCO. Although this model was not conceived for object part segmentation, we find that it can be used as a cost-effective way of obtaining meaningful part segmentations without collecting extra data or using co-part segmentation models that require class-specific hyperparameter tuning, such as SCOPS [18]. Specifically, since VG presents a long tail of rare classes, as in [38] we found it beneficial to first pre-select a small number of representative classes that are widespread across categories (e.g. all land vehicles have wheels, all animals have legs). We set the detection threshold of this model to 0.2 and, for each image category, we only keep semantic classes that appear in at least 25% of the images, which helps eliminate spurious detections. On our data, this leads to a number of semantic classes  $K \approx 10$  per image category (33 across all categories). The full list of semantic classes can be seen in Fig. 5. To deal with potentially overlapping part detections (e.g. the segmentation mask of the door of a car might overlap with a window), the output semantic maps represent probability distributions over classes, where we weight each semantic class proportionally to the object detection score. Additionally, we add an extra class for “no class” (depicted in gray in our figures).

**Mesh templates and remeshing.** We borrow a selection of mesh templates from [28] as well as meshes freely available on the web. In the experiments where we adopt multiple mesh templates, we only use 2–4 meshes per category. An important preliminary step of our approach, which is performed even before the pose estimation step, consists in *remeshing* these templates to align them to a common topology. This has the goal of reducing their complexity (which translates into a speed-up during optimization), removing potential invisible interiors, and enabling efficient batching by making sure that every mesh has the same number of vertices/faces. Additionally, as mentioned in sec. 3.2, remeshing is required for the semi-supervision loss term in the reconstruction model. We frame this task as an optimization problem where we deform a  $32 \times 32$  UV sphere to match the mesh template. More specifically, we render each tem-

plate from 64 random viewpoints at  $256 \times 256$  resolution, and minimize the MSE loss between the rendered deformed sphere and the target template in pixel space ( $\mathcal{L}_{\text{MSE}}$ ). Moreover, we regularize the mesh by adding (i) a smoothness loss  $\mathcal{L}_{\text{flat}}$ , which encourages neighboring faces to have similar normals, (ii) a Laplacian smoothing loss  $\mathcal{L}_{\text{lap}}$  with quad connectivity (i.e. using the topology of the UV map as opposed to that of the triangle mesh), and (iii) an edge length loss  $\mathcal{L}_{\text{len}}$  with quad connectivity, which encourages edges to have similar lengths.  $\mathcal{L}_{\text{flat}}$  and  $\mathcal{L}_{\text{len}}$  are defined as follows:

$$\mathcal{L}_{\text{flat}} = \frac{1}{|E|} \sum_{i,j \in E} (1 - \cos \theta_{ij})^2 \quad (5)$$

$$\mathcal{L}_{\text{len}} = \frac{1}{|UV|} \sum_{i \in U} \sum_{j \in V} \frac{\|\mathbf{v}_{i+1,j} - \mathbf{v}_{i,j}\|_1 + \|\mathbf{v}_{i,j+1} - \mathbf{v}_{i,j}\|_1}{6} \quad (6)$$

where  $E$  is the set of edges,  $\cos \theta_{ij}$  is the cosine similarity between the normals of faces  $i$  and  $j$ , and  $\mathbf{v}_{i,j}$  represents the 3D vertex at the coordinates  $i, j$  of the UV map.

Finally, we weight each term as follows:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + 0.00001 \mathcal{L}_{\text{flat}} + 0.003 \mathcal{L}_{\text{lap}} + 0.01 \mathcal{L}_{\text{len}} \quad (7)$$

Additionally, in the experiments with multiple mesh templates, we add a pairwise similarity loss  $\mathcal{L}_{\text{align}}$  which penalizes large variations of the vertex positions between different mesh templates (only within the same category):

$$\mathcal{L}_{\text{align}} = \frac{1}{N_t^2} \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} \|\mathbf{V}_i - \mathbf{V}_j\|_2 \quad (8)$$

where  $\mathbf{V}_i$  is a matrix that contains the vertex positions of the  $i$ -th mesh template (of shape  $3 \times N_v$ ), and  $N_t$  is the number of mesh templates. This loss term is added to the total loss with weight 0.001. Note that we use a non-squared L2 penalty for this term, which encourages a sparse set of vertices to change between mesh templates.

We optimize the final loss using SGD with momentum (initial learning rate  $\alpha = 0.0001$  and momentum  $\beta = 0.9$ ). We linearly increase  $\alpha$  to 0.0005 over the course of 500 iterations (warm-up) and then exponentially decay  $\alpha$  with rate 0.9999. We stop when the learning rate falls below 0.0001. Additionally, we normalize the gradient before each update.

Fig. 9 shows two qualitative examples of remeshing.



Figure 9. Remeshing of the mesh templates. In this figure we show two demos (one template for *car* and one for *airplane*).

**Pose estimation.** For the silhouette optimization step, we initialize  $N_c = 40$  camera hypotheses per image by uniformly quantizing azimuth and elevation (8 quantization

levels along azimuth and 5 levels along elevation). We optimize each camera hypothesis using Adam [25] with full-matrix preconditioning, where we set  $\beta_1 = 0.9$  and  $\beta_2 = 0.95$ . The implementation of our variant of Adam as well its theoretical justification are described in the next paragraph. We optimize each hypothesis for 100 iterations, with an initial learning rate  $\alpha = 0.1$  which is decayed to 0.01 after the 80th iteration. After each iteration, we re-project quaternions onto the unit ball. As a performance optimization, silhouettes are initially rendered at  $128 \times 128$  resolution, which is increased to  $192 \times 192$  after the 30th iteration and  $256 \times 256$  after the 60th iteration. Finally, in the settings where we prune camera hypotheses, we discard the worst 50% hypotheses as measured by the intersection-over-union (IoU) between projected and target silhouettes. This is performed twice: after the 30th and 60th iteration.

**Algorithm 1** Adam with full-matrix preconditioning. Changes w.r.t. the original algorithm are highlighted.

---

```

1: require  $\alpha$  (step size),  $\beta_1, \beta_2, \epsilon$ 
2: initialize time step  $t \leftarrow 0$ 
3: initialize parameters  $\theta_0$  ( $d$ -dimensional col. vector)
4: initialize first moment  $\mathbf{m}_0 \leftarrow \mathbf{0}$  ( $d$ -dimensional col. vector)
5: initialize second moment  $\mathbf{V}_0 \leftarrow \mathbf{0}$  ( $d \times d$  matrix)
6: repeat
7:    $t \leftarrow t + 1$ 
8:    $\mathbf{g}_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$  ▷ gradient
9:    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$  ▷ first moment
10:   $\mathbf{V}_t \leftarrow \beta_2 \mathbf{V}_{t-1} + (1 - \beta_2) \mathbf{g}_t \mathbf{g}_t^T$  ▷ second moment
11:   $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$  ▷ bias correction
12:   $\hat{\mathbf{V}}_t \leftarrow \mathbf{V}_t / (1 - \beta_2^t)$  ▷ bias correction
13:   $\theta_t \leftarrow \theta_{t-1} - \alpha (\hat{\mathbf{V}}_t + \epsilon \mathbf{I}_d)^{-\frac{1}{2}} \hat{\mathbf{m}}_t$  ▷ update
14: until stopping criterion
15: return  $\theta_t$ 

```

---

**Full-matrix preconditioning.** Adam [25] is an established optimizer for training neural networks. Its use of diagonal preconditioning is an effective trick to avoid storing an  $\mathcal{O}(d^2)$  matrix for the second moments (where  $d$  is the number of learnable parameters), for which a matrix square root and inverse need to be subsequently computed (an extra  $\mathcal{O}(d^3)$  cost for each of the two operations). However, since our goal is to optimize camera parameters, we observe that:

1. Optimizers with diagonal preconditioning are not rotation invariant, i.e. they have some preferential directions that might bias the pose estimation result.
2. Since each camera hypothesis comprises only 8 parameters, inverting an  $8 \times 8$  matrix has a negligible cost.

Using a rotation invariant optimizer such as SGD (with or without momentum) is a more principled choice as it addresses the first observation. However, based on our second observation, we take the best of both worlds and modify

Adam to implement full-matrix preconditioning. This only requires a trivial modification to the original implementation, which we show in [alg. 1](#) (changes w.r.t. the original algorithm are highlighted in green).

**Semantic template inference.** As mentioned in [sec. 3.1](#), the goal of this step is to infer a 3D semantic template for each mesh template, given an initial (untextured) mesh template, the output of the silhouette optimization step, and a collection of 2D semantic maps. Recapitulating from [sec. 3.1](#), we solve the following optimization problem:

$$\mathcal{L}_i = \|\mathcal{R}(\mathbf{V}_{\text{tpl}}, \mathbf{F}_{\text{tpl}}, \mathbf{C}_{\text{tpl}}; \mathbf{q}_i, \mathbf{t}_i, s_i, z_{0i}) - \mathbf{C}_i\|^2 \quad (9)$$

$$\mathbf{C}_{\text{tpl}}^* = \min_{\mathbf{C}_{\text{tpl}}} \frac{1}{N_{\text{top}}} \sum_i \mathcal{L}_i \quad (10)$$

Conceptually, our goal is to learn a shared semantic template (parameterized using vertex colors) that averages all 2D semantic maps in vertex space. We propose the following closed-form solution which uses the gradients from the differentiable renderer and requires only a single pass through the dataset:

$$\mathbf{A} = \sum_i \nabla_{\mathbf{C}_{\text{tpl}}}(\mathcal{L}_i) \quad (11)$$

$$(\mathbf{C}_{\text{tpl}}^*)_k = \frac{\epsilon + \mathbf{a}_k}{K\epsilon + \sum_j \mathbf{a}_j} \quad (12)$$

where  $\mathbf{A}$  is an accumulator matrix that has the same shape as the  $\mathbf{C}_{\text{tpl}}$  (the vertex colors), and  $\epsilon$  is a small *additive smoothing* constant that leads to a uniform distribution on vertices that are never rendered (and thus have no gradient). This operation can be regarded as projecting the 2D object-part semantics onto the mesh vertices and computing a color histogram on each vertex. We show a sample illustration in [Fig. 10](#).

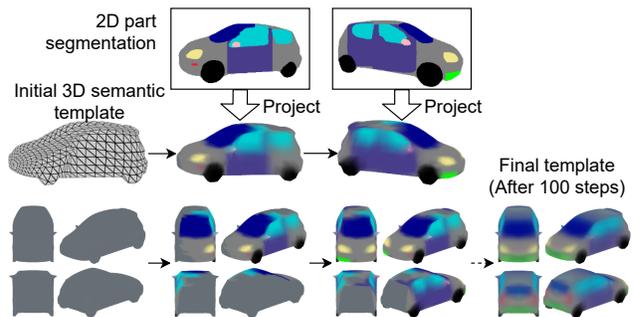


Figure 10. Semantic template inference, starting from an untextured 3D mesh template (left-to-right progression). In this figure we show a demo with two sample images, and the final result using the top 100 images as measured by the IoU.

In [section sec. 3.1](#) we explained that we compute the semantic template using the top  $N_{\text{top}} = 100$  images as mea-

sured by the IoU, among those that passed the ambiguity detection test ( $v_{agr} < 0.3$ ). To further improve the quality of the inferred semantic templates, we found it beneficial to add an additional filter where we only select poses whose cosine distance is within 0.5 (i.e. 45 degrees) of the left/right side. Objects observed from the left/right side are intrinsically unambiguous, since there is no complementary pose that results in the same silhouette. Therefore, we favor views that are close to the left/right as opposed to the front/back or top/bottom, which are the most ambiguous views. Note that this filter is only used for the semantic template inference step.

**Generative model.** We train the single-category reconstruction networks (setting **A**) for 130k iterations, with a batch size of 32, and on a single GPU. The multi-category model (setting **B**) is trained for 1000 epochs, with a total batch size of 128 across 4 GPUs, using synchronized batch normalization. In both settings, we use Adam [25] (the original one, not our variant with full-matrix preconditioning) with an initial learning rate of 0.0001 which is halved at 1/4, 1/2, 3/4 of the training schedule. For the GAN, we use the same hyperparameters as [39], except in the multi-category model (setting **B**), which is trained with a batch size of 64 instead of the default 32. Furthermore, in setting **B**, and for both models (reconstruction and GAN), we equalize classes during mini-batch sampling. This is motivated by the large variability in the amount of training images, as explained in sec. 4.1, and as can also be seen in Table 3. Finally, as in [39, 20, 10, 29], we force generated meshes to be left/right symmetric.

**Semantic mesh generation.** In the setting where we generate a 3D mesh from a semantic layout in UV space, we modify the generator architecture of [39]. Specifically, we replace the input linear layer (the one that projects the latent code  $z$  onto the first  $8 \times 8$  convolutional feature map) with four convolutional layers. These progressively down-sample the semantic layout from  $128 \times 128$  down to  $8 \times 8$  (i.e. each layer has stride 2). The first layer takes as input a *one-hot* semantic map (with  $K$  semantic channels) and yields 64 output channels (128, 256, 512 in the following layers). In these 4 layers, we use Leaky ReLU activations (slope 0.2), spectral normalization, but no batch normalization. We leave the rest of the network unchanged. In this model, we also found it necessary to fine-tune the batch normalization statistics prior to evaluation, which we do by running a forward pass over the entire dataset on the *running average* model. As for the discriminator, we simply resize the semantic map as required and concatenate it to the input.

## A.2. Additional results

**Pose estimation.** In Fig. 11, we provide more insight into the *geodesic distance* metric, which measures the cosine

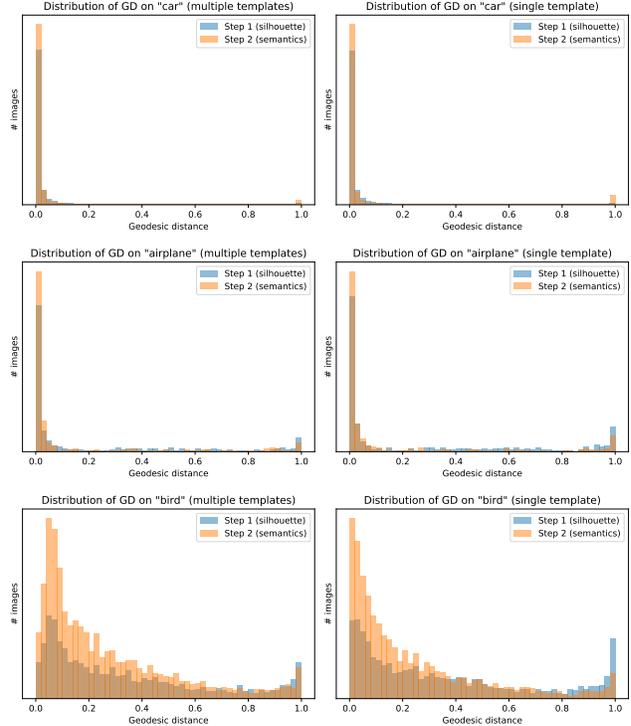


Figure 11. Distribution of pose estimation errors on *car*, *airplane*, and *bird*. We compare settings where we use multiple mesh templates (left) and a single template (right).

distance between the rotations predicted by our approach (sec. 3.1) and SfM rotations. In particular, as opposed to the results presented in Table 1 (which shows only the average), here we show the full distribution of errors. A distance of 0 means that the two rotations match exactly, whereas a distance of 1 (maximum value) means that the rotations are rotated by 180 degrees from one another. On the analyzed classes (*car*, *airplane*, and *bird*, for which we have SfM poses), we can generally observe a bimodal distribution: a majority of images where pose estimation is correct, i.e. the GD is close to zero, and a small cluster of images where the GD is close to one. This is often the case for ambiguities: for instance, in cars we sometimes observe a front/back confusion. As expected, exploiting semantics (step 2) mitigates this issue and increases the amount of available images (this is particularly visible on *bird*). We also note that, for rigid objects such as *car* and *airplane*, the distribution is more peaky, whereas for *bird* the tail of errors is longer, most likely because pose estimation is more ill-defined for articulated objects.

**Qualitative results.** We show extra qualitative results in Fig. 14. In particular, we render each generated mesh from two random viewpoints and showcase the associated texture and wireframe mesh. Additionally, in Fig. 12 we show the most common failure cases across categories. We can



Figure 12. Failure cases for a variety of categories.

identify some general patterns: for instance, in vehicles we sometimes observe incoherent textures (this is particularly visible in *truck* due to the small size of this dataset). On animals, as mentioned, we observe occasional failures to model facial details, merged/distorted legs, and more rarely, mesh distortions. To some extent, these issues can be mitigated by sampling from the generator using a lower truncation threshold (we use  $\sigma = 1.0$  in our experiments), at the expense of sample diversity.

**Semantic templates.** Fig. 13 shows the full set of learned semantic templates for every category. Most results are coherent, although we observe a small number of failure cases, e.g. in *truck* one or two templates are mostly empty and are thus ineffective for properly resolving ambiguities. This generally happens when the templates have too few images assigned to them and explains why the multi-template setting does not consistently outperform the single-template setting.

**Demo video.** The supplementary material includes a video where we show additional qualitative results. First, we showcase samples generated by our models in setting **A** and explore the latent space of the generator. Second, we analyze the latent space of the model trained to generate multiple classes (setting **B**), and discover interpretable directions in the latent space, which can be used to control shared aspects between classes (e.g. lighting, shadows). We also interpolate between different classes while keeping the latent space fixed, and highlight that style is preserved during interpolation. Finally, we showcase a setting where we generate a mesh from a hand-drawn semantic layout in UV space, similar to Fig. 8.

Class	Synsets	Raw images	Valid instances
Motorbike	n03790512, n03791053, n04466871	4037	1351
Bus	n04146614, n02924116	2641	1190
Truck	n03345487, n03417042, n03796401	3187	1245
Car	n02814533, n02958343, n03498781, n03770085, n03770679, n03930630, n04037443, n04166281, n04285965	12819	4992
Airplane	n02690373, n02691156, n03335030, n04012084	5208	2540
Sheep	n10588074, n02411705, n02413050, n02412210	4682	864
Elephant	n02504013, n02504458	3927	1434
Zebra	n02391049, n02391234, n02391373, n02391508	5536	1753
Horse	n02381460, n02374451	2589	664
Cow	n01887787, n02402425	2949	861
Bear	n02132136, n02133161, n02131653, n02134084	6745	2688
Giraffe	n02439033	1256	349

Table 3. Synsets and summary statistics for our ImageNet data. For each category, we report the number of raw images in the dataset, and the number of extracted object instances that have passed our quality checks (size, truncation, occlusion).

### A.3. Dataset information

For our experiments on ImageNet, we adopt the *synsets* specified in Table 3. Since some of our required synsets are not available in the more popular ImageNet1k, we draw all of our data from the larger ImageNet22k set.

### A.4. Negative results

To guide potential future work in this area, we provide a list of ideas that we explored but did not work out.

**Silhouette optimization.** For the silhouette optimization step with multiple templates, before reaching our current formulation, we explored a range of alternatives. In particular, we tried to smoothly interpolate between multiple meshes by optimizing a set of interpolation weights along with the camera parameters. This yielded inconsistent results across categories, which convinced us to work with a “discrete” approach as opposed to a smooth one. We then tried a reinforcement learning approach inspired by multi-armed bandits: we initialized each camera hypothesis with a random mesh template, and used a UCB (upper confidence bound) selection algorithm to select the optimal mesh template during optimization. This led to slightly worse results than interpolation. Finally, we reached our current formulation, where we simply replicate each camera hypothesis and optimize the different mesh templates separately. We adopt pruning to make up for the increase in computation time.

**Re-optimizing poses multiple times.** In our current formulation, after the semantic template inference step, we use the semantic templates to resolve ambiguities, but there is no further optimization involved. Naturally, we explored the idea of repeating the silhouette optimization step using semantic information. However, we were unable to get this step to work reliably, even after attempting with multiple

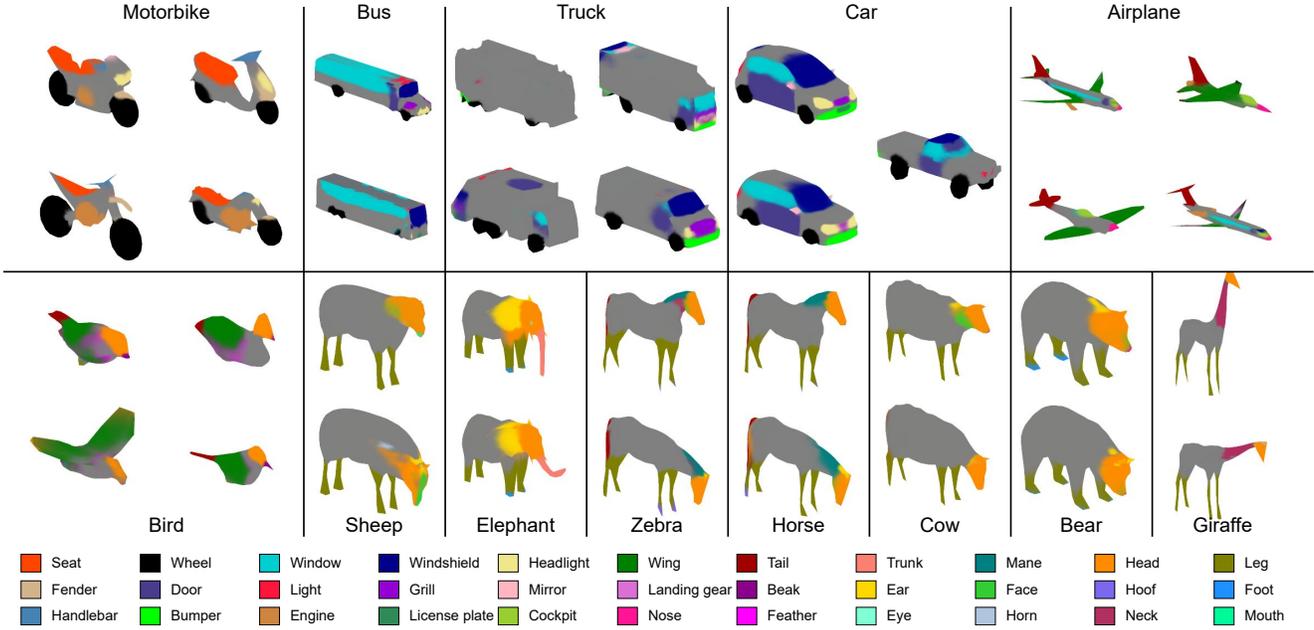


Figure 13. Visualization of *all* the learned 3D semantic templates (2–4 per category). While most results are as expected, the figure highlights some failure cases, e.g. in *truck* some templates have very few images assigned to them, which leads to incoherent semantics.

renderers (we tried both with DIB-R [4] and SoftRas [32]). We generally observed that the color gradients are too uninformative for optimizing camera poses, even after trying to balance the different components of the gradient (silhouette and color). We believe this is a fundamental issue related to the non-convexity of the loss landscape, which future work needs to address. We also tried to smooth out the rendered images prior to computing the MSE loss, without success.

**Remeshing.** Since target 3D vertices are known in this step, we initially tried to use a 3D chamfer loss to match the mesh template. This, however, led to artifacts and merged legs in animals, and was too sensitive to initialization. We found it more reliable to use a differentiable render with silhouette-based optimization.



Figure 14. Additional qualitative results. We show three examples per category. Each example is rendered from two random views, and the corresponding texture/wireframe mesh is also shown.