Supplementary Material for *Leveraging 2D Data to Learn Textured 3D Mesh Generation*

Paul Henderson, Vagia Tsiminaki and Christoph H. Lampert

S1 Single-image 3D reconstruction learnt from natural images

Our encoder network yields a variational posterior distribution on the latent variables explaining a given image (see main paper, Section 5), while the shape and color decoders $\mathrm{dec}_{\mathrm{shape}}$ and $\mathrm{dec}_{\mathrm{color}}$ map from the latent variables to a textured 3D mesh (see main paper, Section 3). Using these trained components together, we can perform single-image 3D reconstruction, in spite of the model never having received any 3D supervision, nor even segmentation masks in (NO-MASK) setting. This comes 'for free' as a result of training the generative model; in particular, we did *not* tune our models for reconstruction quality.

In this section, we present qualitative examples of reconstructing cars and birds from natural images. Fig. S1 and Fig. S2 show examples in setting (MASK), for cars and birds respectively. Note that the background used to illustrate the reconstruction for birds is generated from the original image, with all points inside the (ground-truth) bird mask replaced by the nearest non-masked pixel. Similarly, Fig. S3 and Fig. S4 show examples in setting (NO-MASK).

S2 Additional samples for natural image classes

In this section, we present random (uncurated) samples generated by our model for the classes car (Fig. S5 and Fig. S6) and bird (Fig. S7 and Fig. S8), in settings (MASK) and (NO-MASK). See the main paper, Section 6.2, for discussion of results.

S3 Additional samples for ShapeNet classes

In this section, we present random (uncurated) samples generated by our model for the four ShapeNet classes car (Fig. S9 and Fig. S10), chair (Fig. S11, Fig. S12 and Fig. S13), airplane (Fig. S14, Fig. S15 and Fig. S16), and sofa (Fig. S17, Fig. S18 and Fig. S19), in settings (MASK) and (NO-MASK). See the main paper, Section 6.1, for discussion of results.

S4 Network architectures

In this section, we describe the neural network architectures for each component of our model.

Notation. We use the following notation for network layers:

- Convolutional layers are denoted by Conv(channels, filter size); stride is one unless otherwise specified
- Densely-connected layers are denoted by Dense(channels)
- Bilinear upsampling layers are denoated by Upsampling(factor)
- Reshape(shape) denotes reshaping the input tensor to the given shape
- $\oplus$ denotes vector concatenation
- σ denotes the logistic sigmoid function

When the input to a layer is not just the output of the previous layer, we indicate this input in a second pair of parentheses.

Encoders. The encoders consist of a shared CNN that extracts features from the input image, followed by densely-connected layers operating on the resulting features to give predictions for each latent variable. The feature extractor is as follows (all convolution/dense layers use relu activation and group-normalization):

- *input: RGB image*
- $\mathrm{Conv}(32, 3\times 3, \mathrm{stride}=2)$
- $\mathrm{Conv}(64, 3\times 3)$
- $\mathrm{MaxPool}(2\times 2)$

- $\text{Conv}(96, 3 \times 3)$
- $\text{MaxPool}(2 \times 2)$
- $\text{Conv}(128, 3 \times 3)$
- $\text{MaxPool}(2 \times 2)$
- $\text{Conv}(128, 4 \times 4)$
- $\text{Dense}(128)$
- *output: 128D feature vector* $\mathbf{f}$

The shape encoder is as follows:

- *input: 128D feature vector* $\mathbf{f}$
- $\text{Dense}(32 \times 2)$
- *output: mean and stddev of 32D shape embedding* $\mathbf{z}_{\text{shape}}$*, latter with softplus activation*

The texture encoder is as follows:

- *input: 128D feature vector* $\mathbf{f}$ *and vector* $\mathbf{c}$ *of mean pixel colors clipping each face*
- $\text{Dense}(96, \text{activation} = \text{relu}, \text{group-normalization})(\mathbf{c}) \oplus \mathbf{f}$
- $\text{Dense}(128 \times 2)$
- *output: mean and stddev of 128D texture embedding* $\mathbf{z}_{\text{color}}$*, latter with softplus activation*

The background encoder is as follows:

- *input: 128D feature vector* $\mathbf{f}$
- $\text{Dense}(64, \text{activation} = \text{relu}, \text{group-normalization})$
- $\text{Dense}(16 \times 2)$
- *output: mean and stddev of 16D background embedding* $\mathbf{z}_{\text{bg}}$*, latter with softplus activation*

The pose encoder is as follows:

- *input: 128D feature vector* $\mathbf{f}$
- $\text{Dense}(5)$
- *output: 2D offset in xz-plane; 3D log-scale*

Decoders. The decoders consist of densely-connected networks, taking the latent variables as input. The shape decoder $\text{dec}_{\text{shape}}$ is as follows:

- *input: 32D shape embedding* $\mathbf{z}_{\text{shape}}$
- $\text{Dense}(32, \text{activation} = \text{elu})$
- $\text{Dense}(3N_V)$
- $\text{Reshape}(N_V, 3)$
- *output: 3D offset vectors to be added to each of the* N_V *vertices of the base mesh*

The texture decoder $\text{dec}_{\text{color}}$ is as follows:

- *input: 128D texture embedding* $\mathbf{z}_{\text{color}}$ *and 32D shape embedding* $\mathbf{z}_{\text{shape}}$
- $\text{Dense}(128, \text{activation} = \text{elu})(\mathbf{z}_{\text{color}} \oplus \mathbf{z}_{\text{shape}}) + \mathbf{z}_{\text{color}}$
- $\text{Dense}(192, \text{activation} = \text{elu})$
- $\text{Dense}(3N_F)$
- $\text{Reshape}(N_F, 3) \,/\, 10 + \frac{1}{2}$
- *output: RGB colors for each of the* N_F *faces of the mesh*

The background decoder, used only in setting (NO-MASK), is as follows (all convolution layers use elu activation, except the last):

- *input: 16D background embedding*
- $\text{Reshape}(1 \times 1 \times 16)$
- $\text{Upsample}(4\times)$
- $\text{Conv}(64, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(32, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(16, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\sigma(\text{Conv}(3, 3 \times 3) \,/\, 2)$
- $\text{Upsample}(4\times \text{ or } 6\times)$
- *output: RGB background image*

Baseline VAE. The encoder for the baseline VAE uses the same feature extractor as the main model. To convert the features to the mean and variance of the latent variables, the following architecture is used:

- *input: 128D feature vector* $\mathbf{f}$
- $\text{Dense}(160 \times 2)$
- *output: mean and stddev of 160D image embedding, latter with softplus activation*

The decoder is as follows (all layers use elu activation, except the last):

- *input: 160D image embedding*
- $\text{Dense}(256)$
- $\text{Reshape}(4 \times 4 \times 16)$
- $\text{Conv}(128, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(64, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(32, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(24, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(16, 3 \times 3)$
- $\text{Upsample}(2\times)$
- $\text{Conv}(3, 3 \times 3) + \frac{1}{2}$
- *output: RGB image*

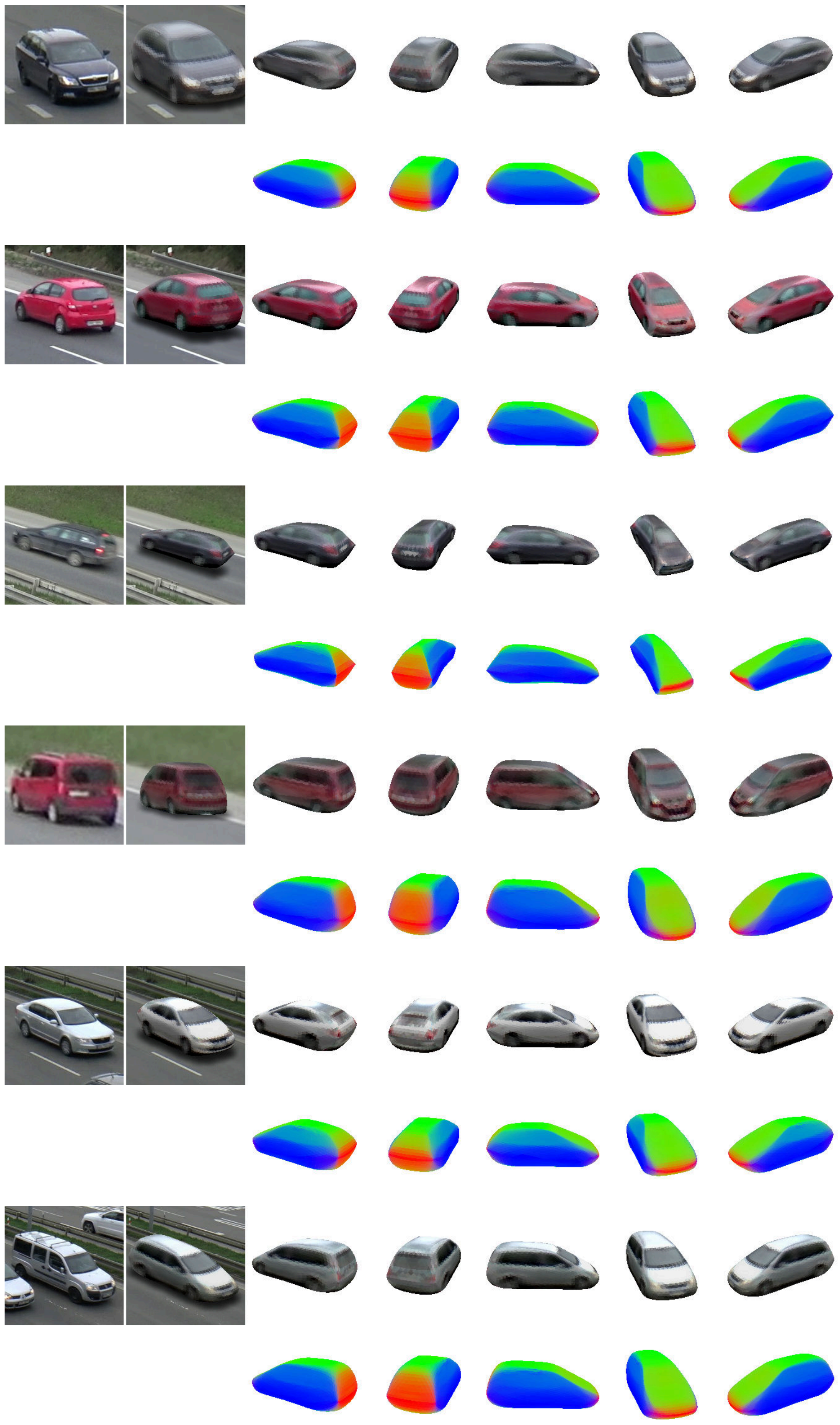

Figure S1: Examples of reconstructions of cars, in setting (MASK). The left-hand image is the input to our model; the next is our reconstruction, rendered over the ground-truth background; the remaining five columns are different views of the reconstructed instance, with normal-maps below

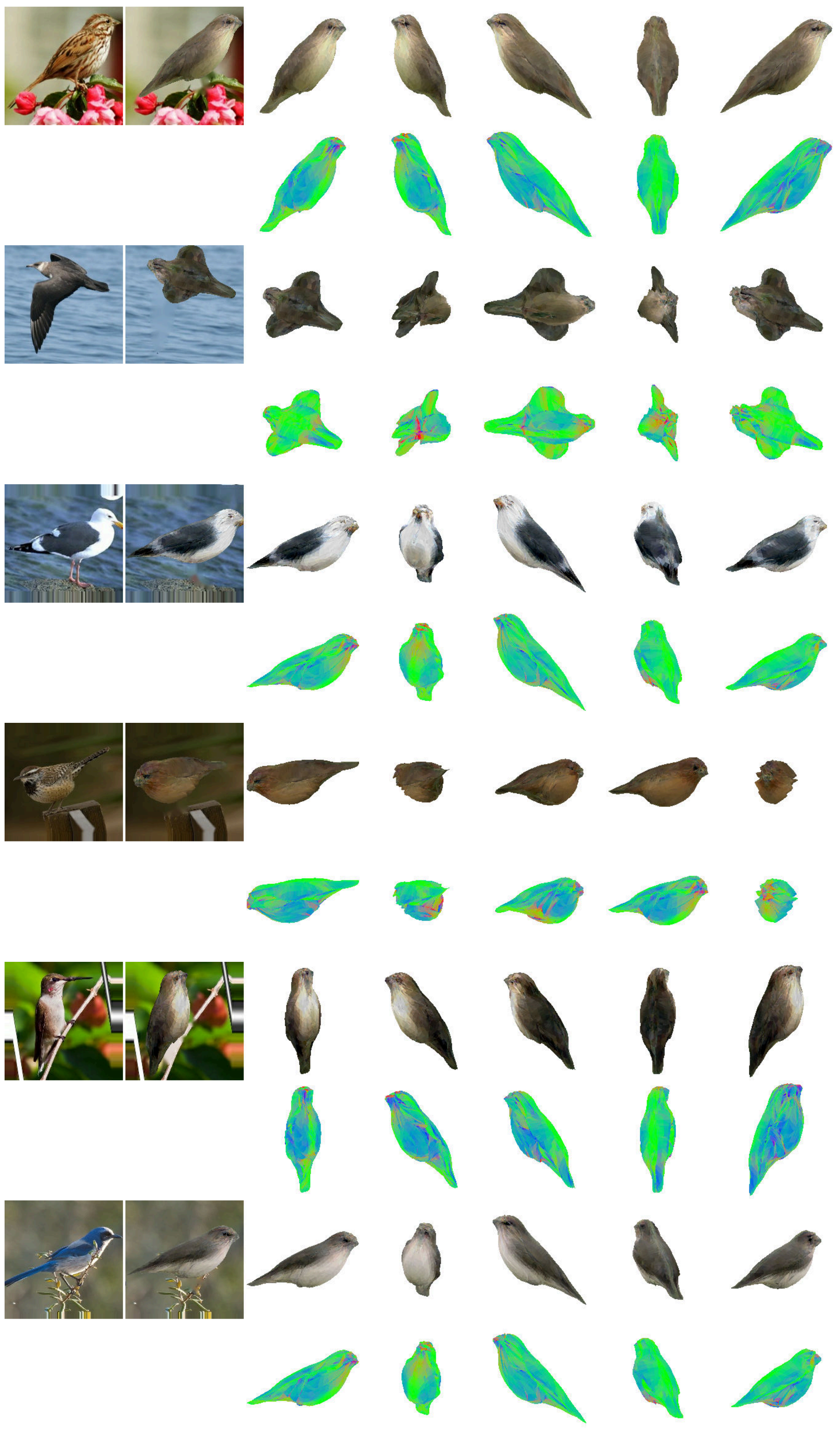

Figure S2: Examples of reconstructions of birds, in setting (MASK). The left-hand image is the input to our model; the next is our reconstruction, rendered over a pseudo-background (see text); the remaining five columns are different views of the reconstructed instance, with normal-maps below

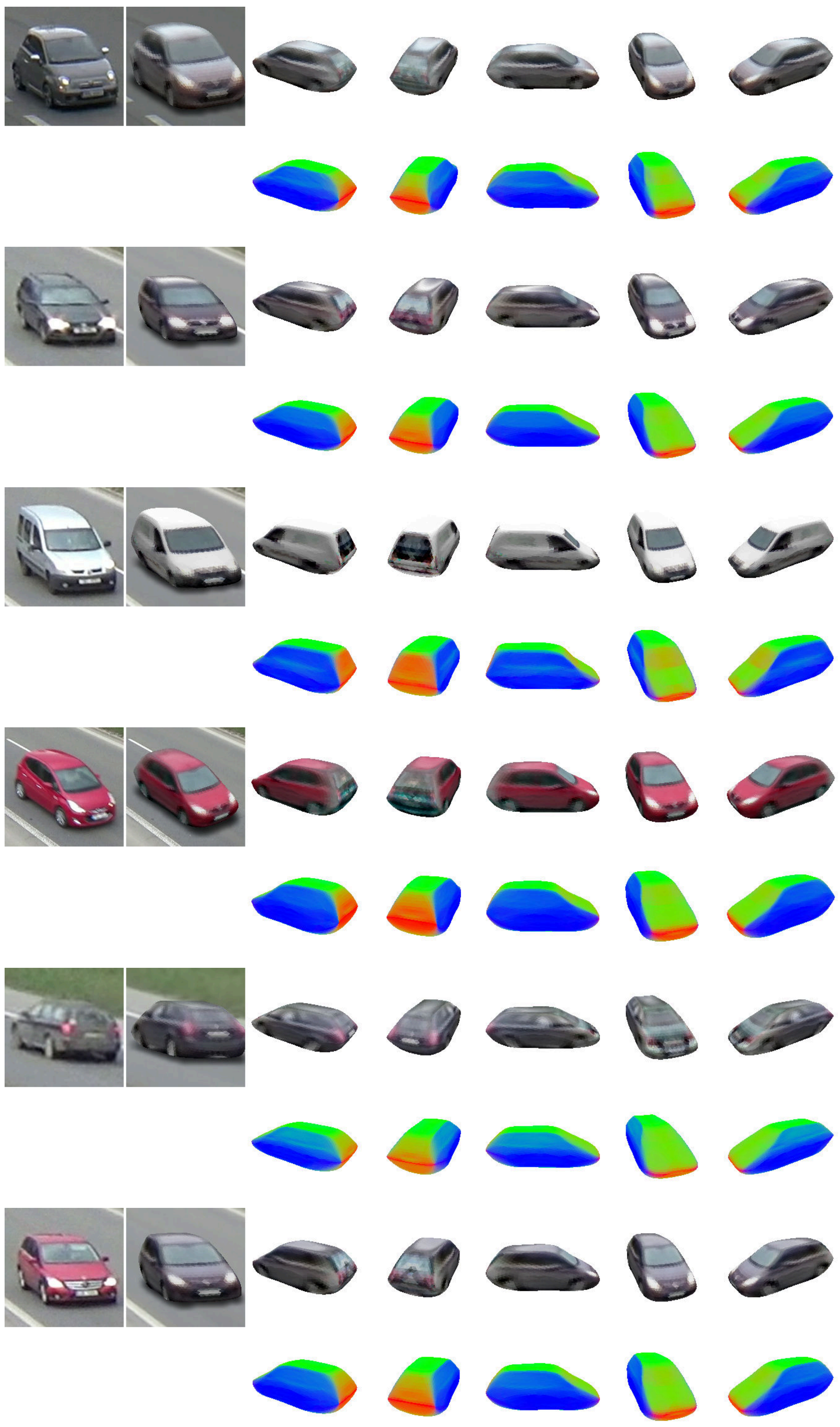

Figure S3: Examples of reconstructions of cars, in setting (NO-MASK). The left-hand image is the input to our model; the next is our reconstruction, rendered over the ground-truth background; the remaining five columns are different views of the reconstructed instance, with normal-maps below

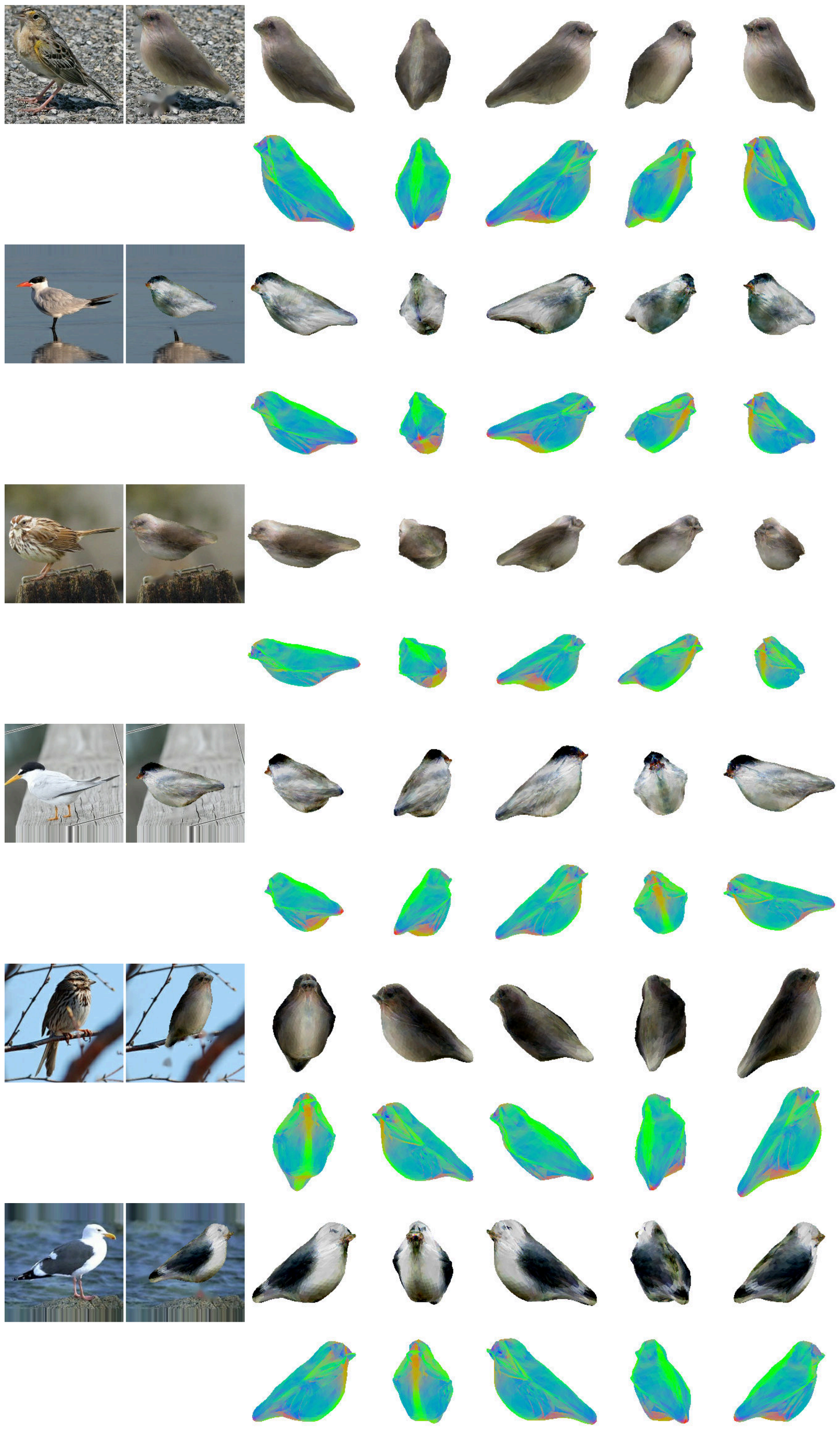

Figure S4: Examples of reconstructions of birds, in setting (NO-MASK). The left-hand image is the input to our model; the next is our reconstruction, rendered over a pseudo-background (see text); the remaining five columns are different views of the reconstructed instance, with normal-maps below

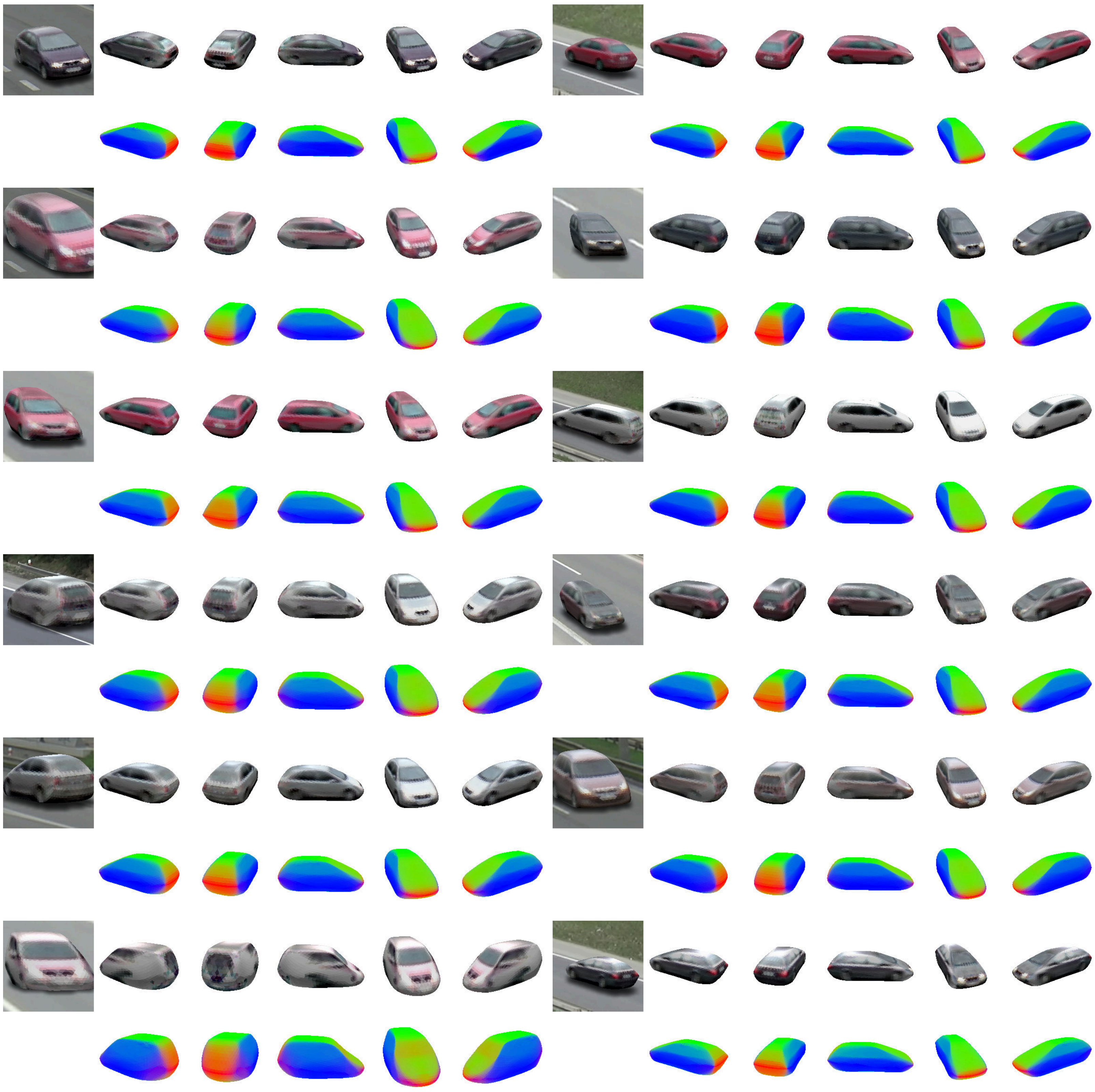

Figure S5: Examples of cars generated by our model, in setting (MASK).

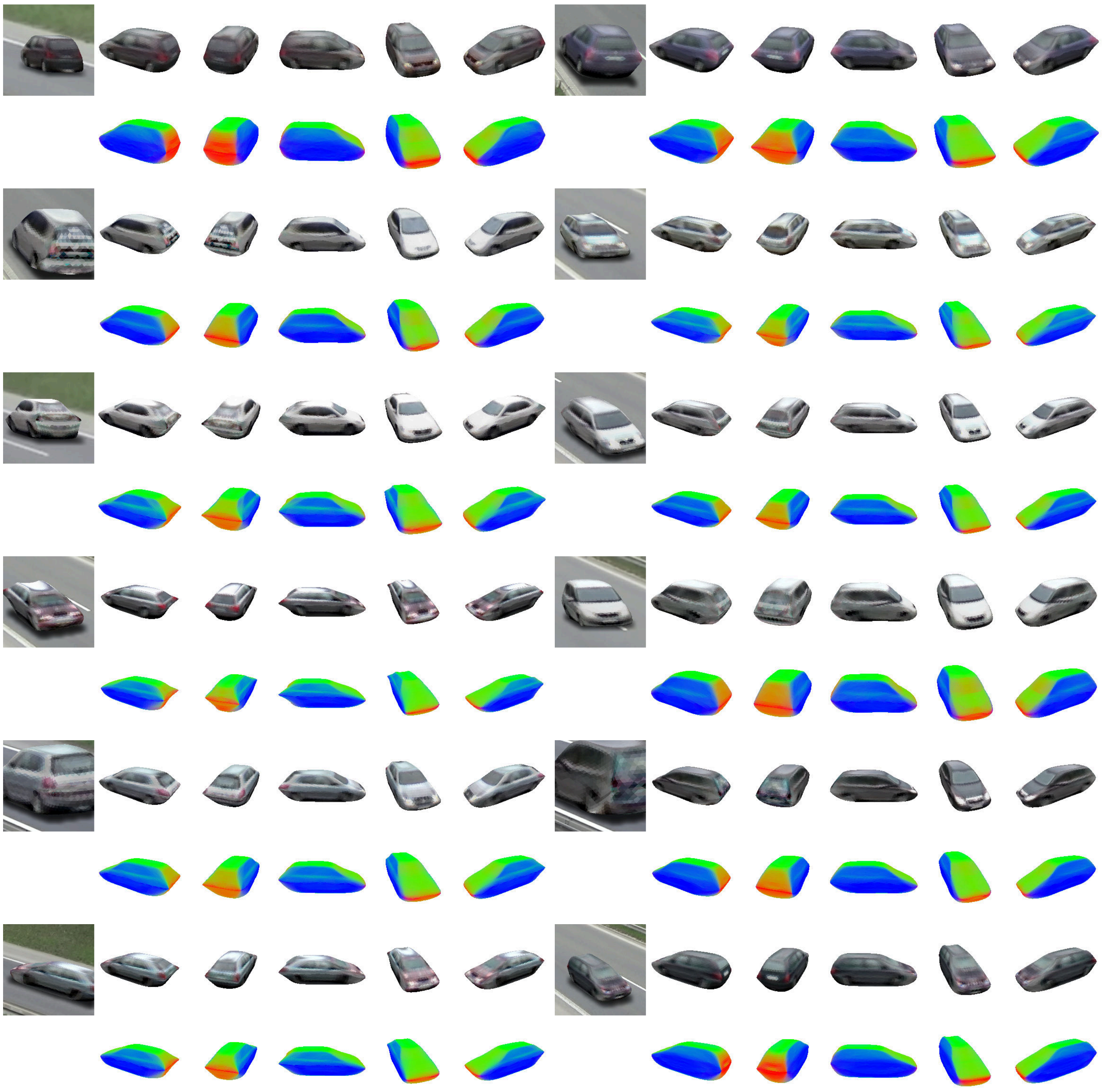

Figure S6: Examples of cars generated by our model, in setting (NO-MASK).

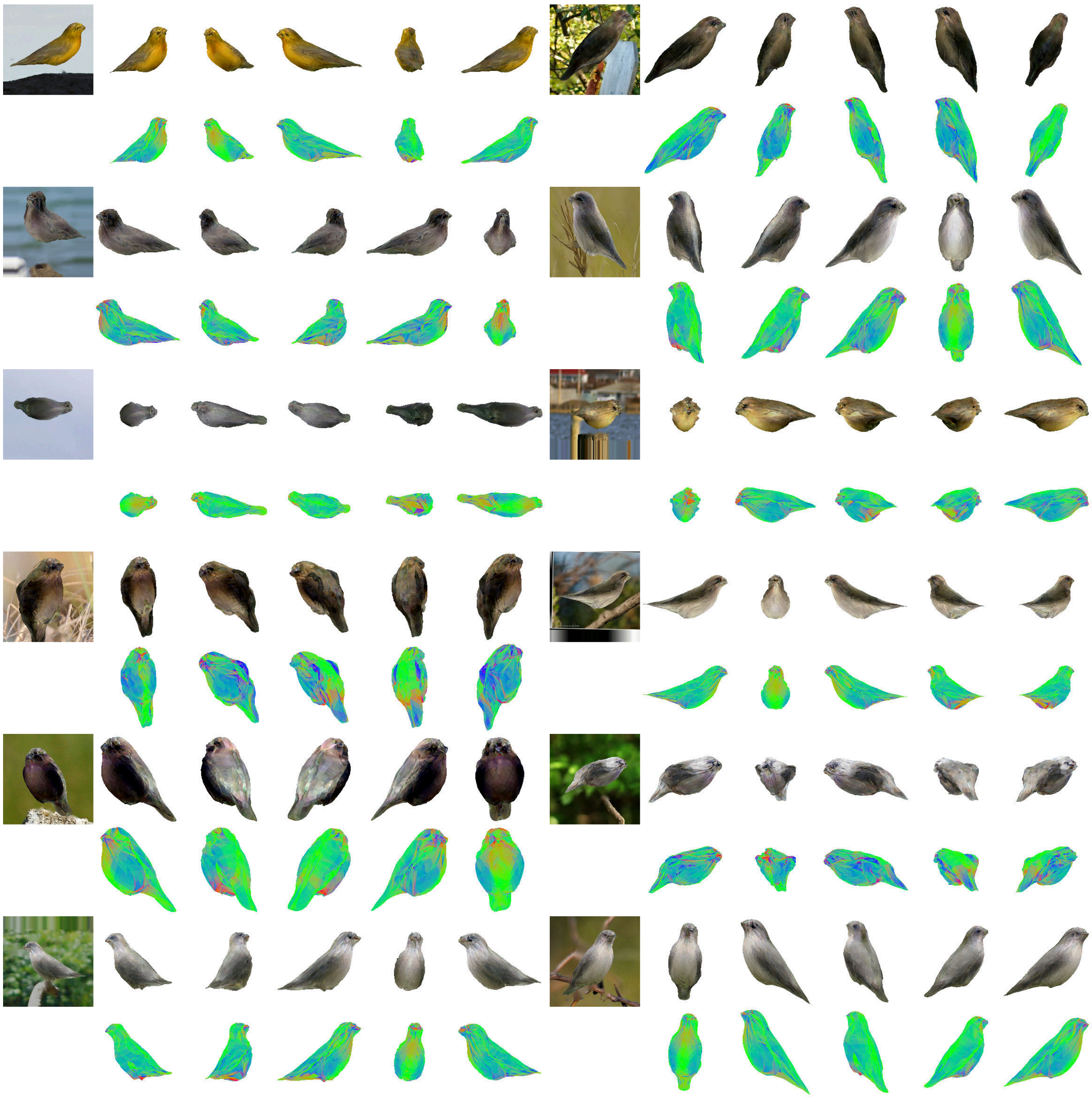

Figure S7: Examples of birds generated by our model, in setting (MASK).

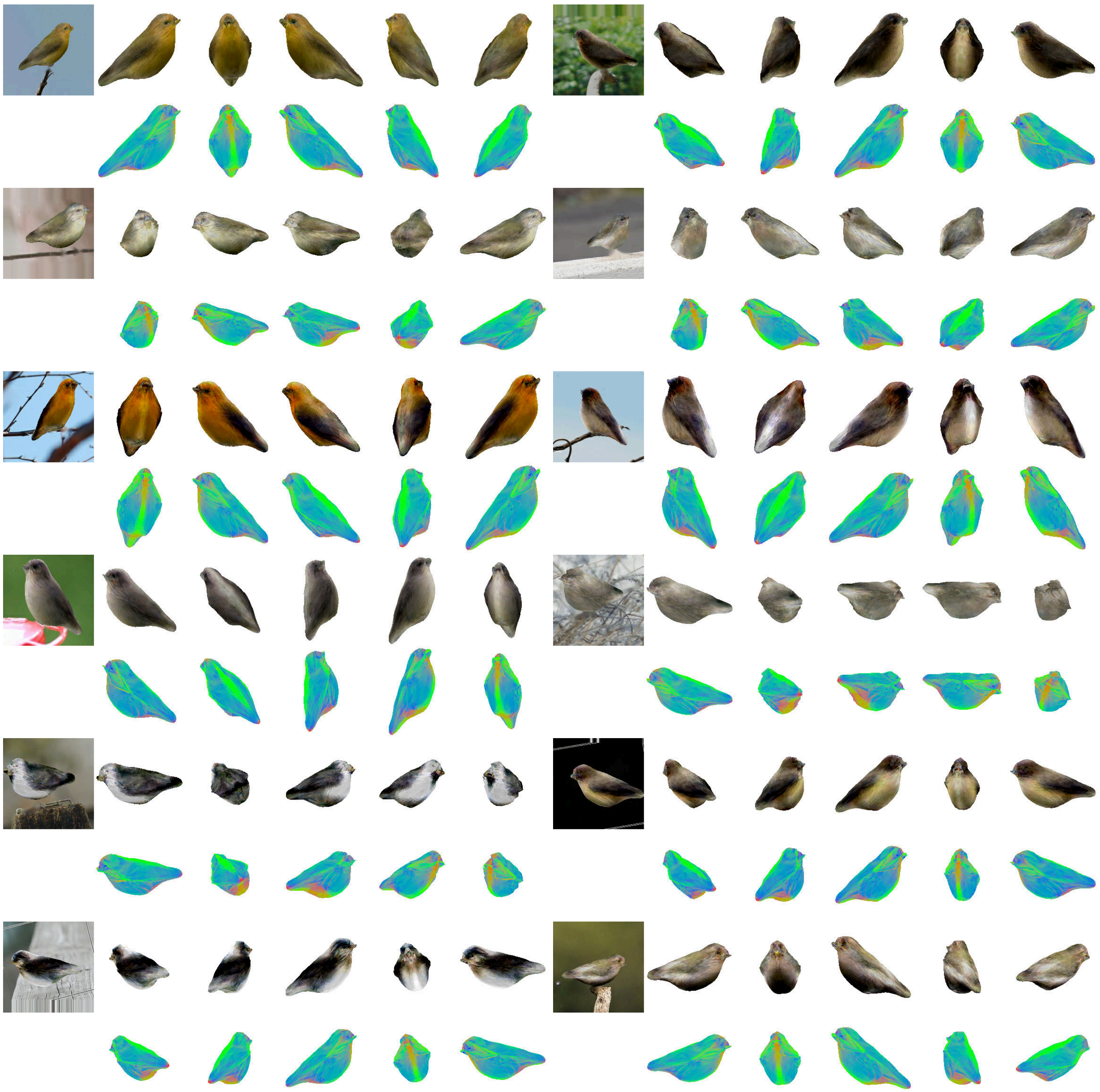

Figure S8: Examples of birds generated by our model, in setting (NO-MASK).

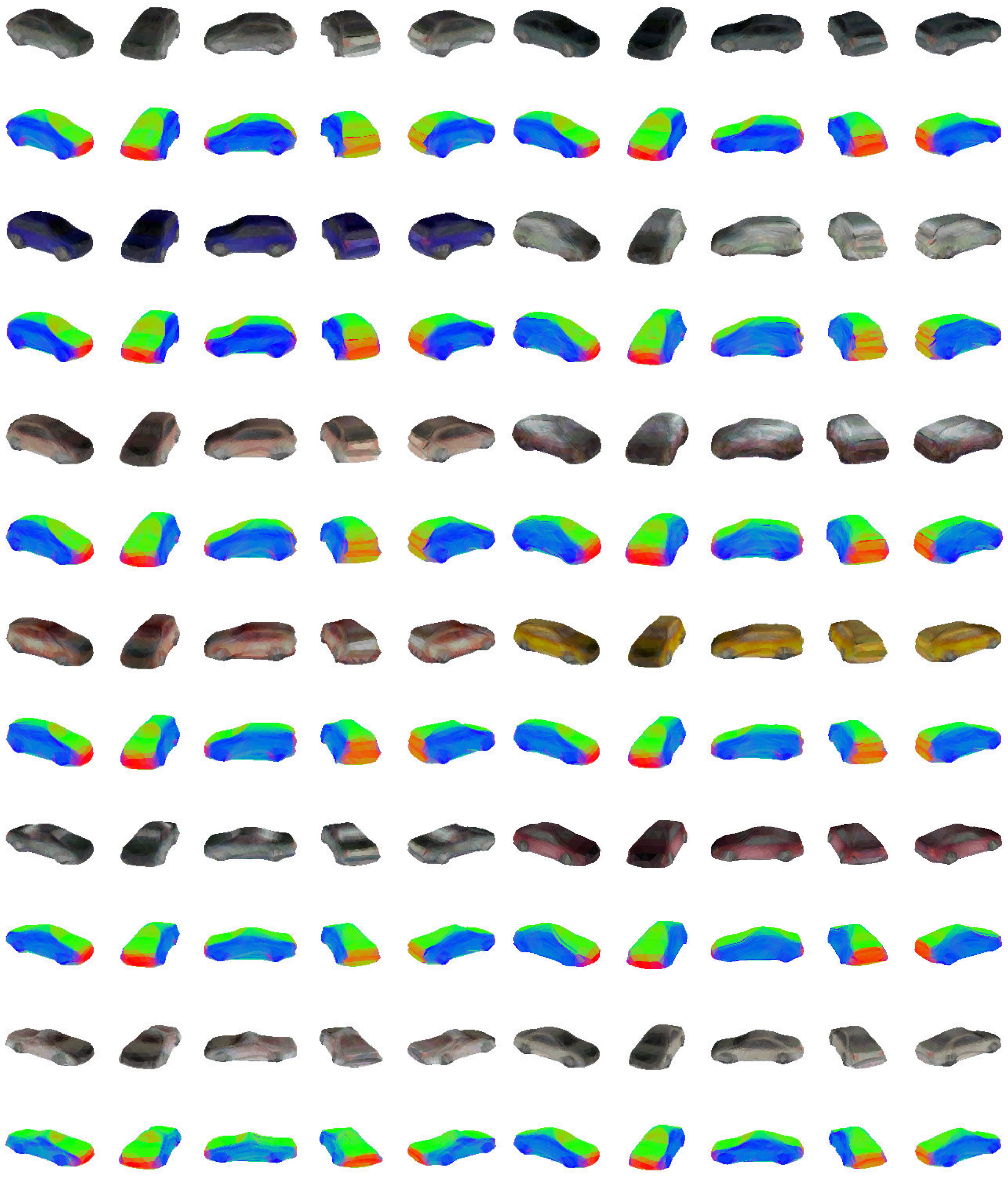

Figure S9: Examples of cars generated by our model, in setting (MASK) with parametrization (DENSE).

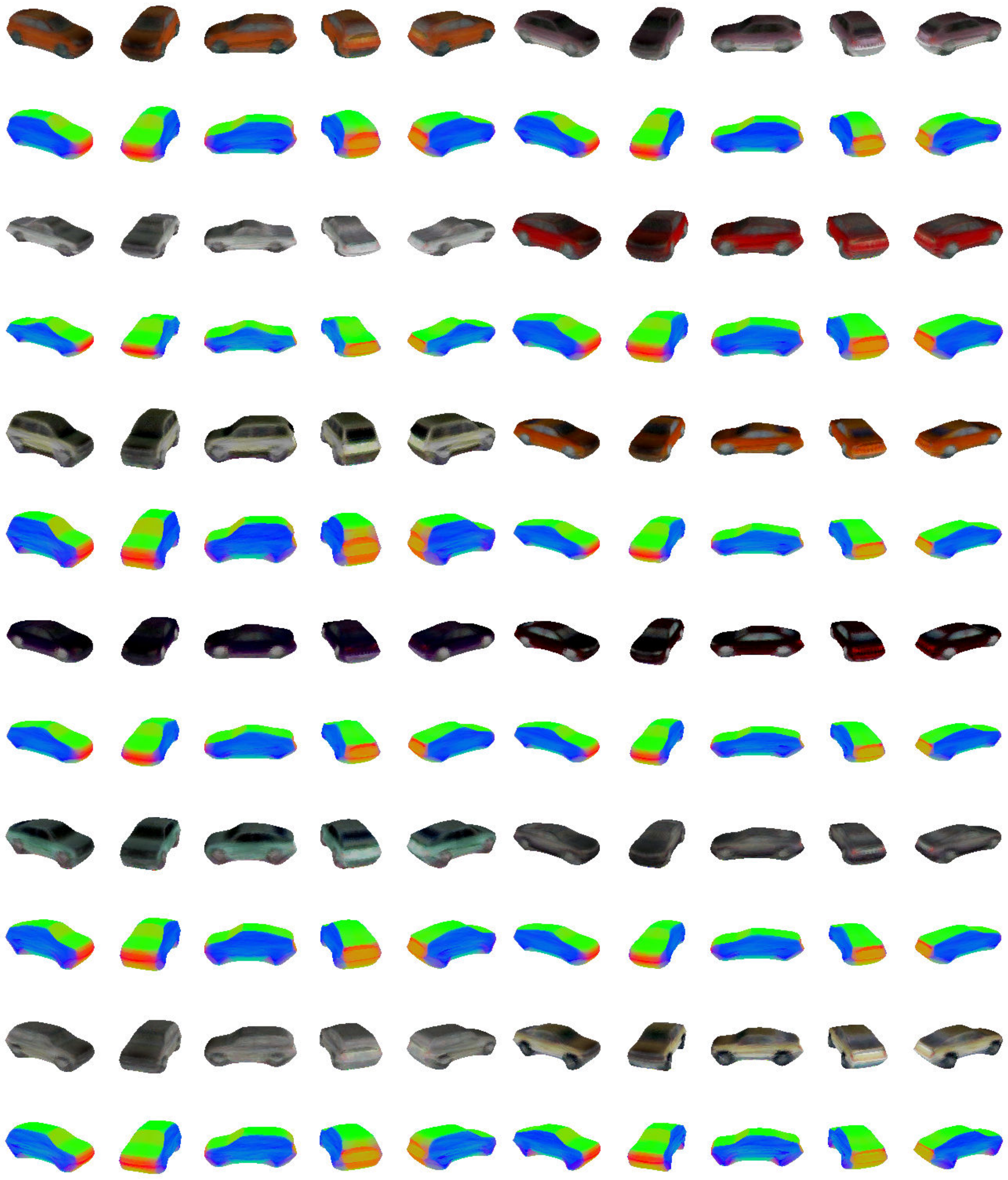

Figure S10: Examples of cars generated by our model, in setting (NO-MASK) with parametrization (DENSE).

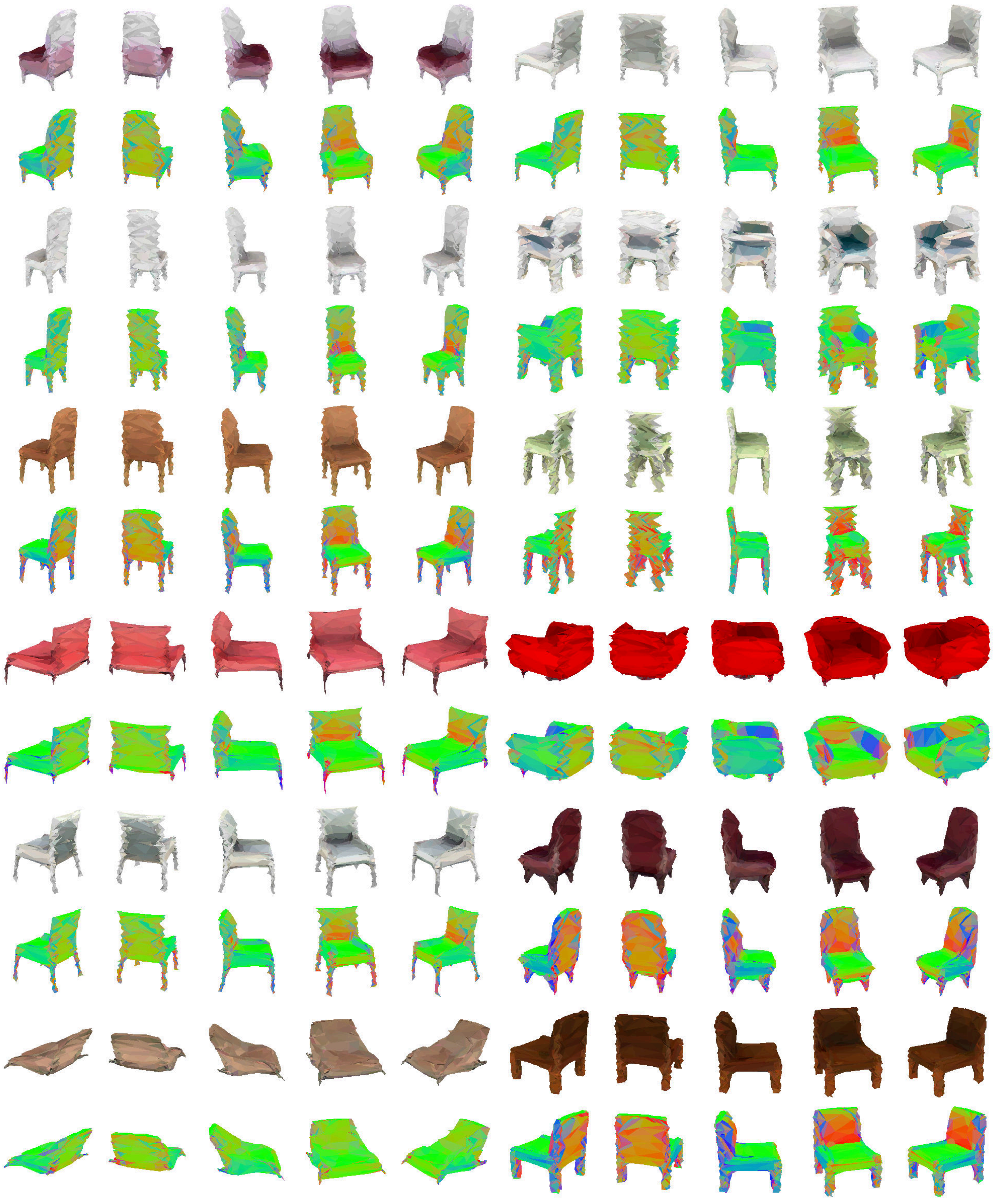

Figure S11: Examples of chairs generated by our model, in setting (MASK) with parametrization (DENSE).

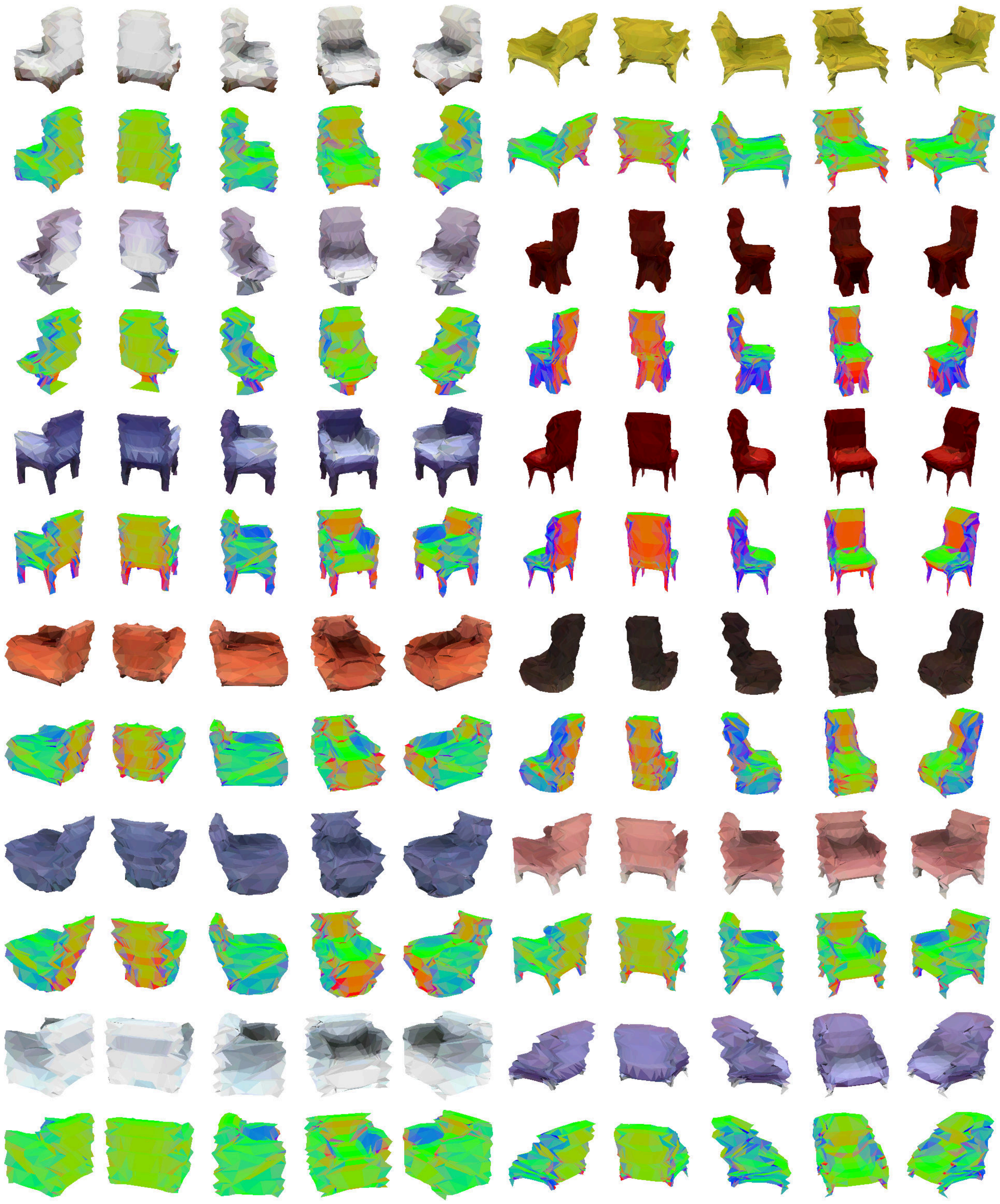

Figure S12: Examples of chairs generated by our model, in setting (MASK) with parametrization (PUSHING).

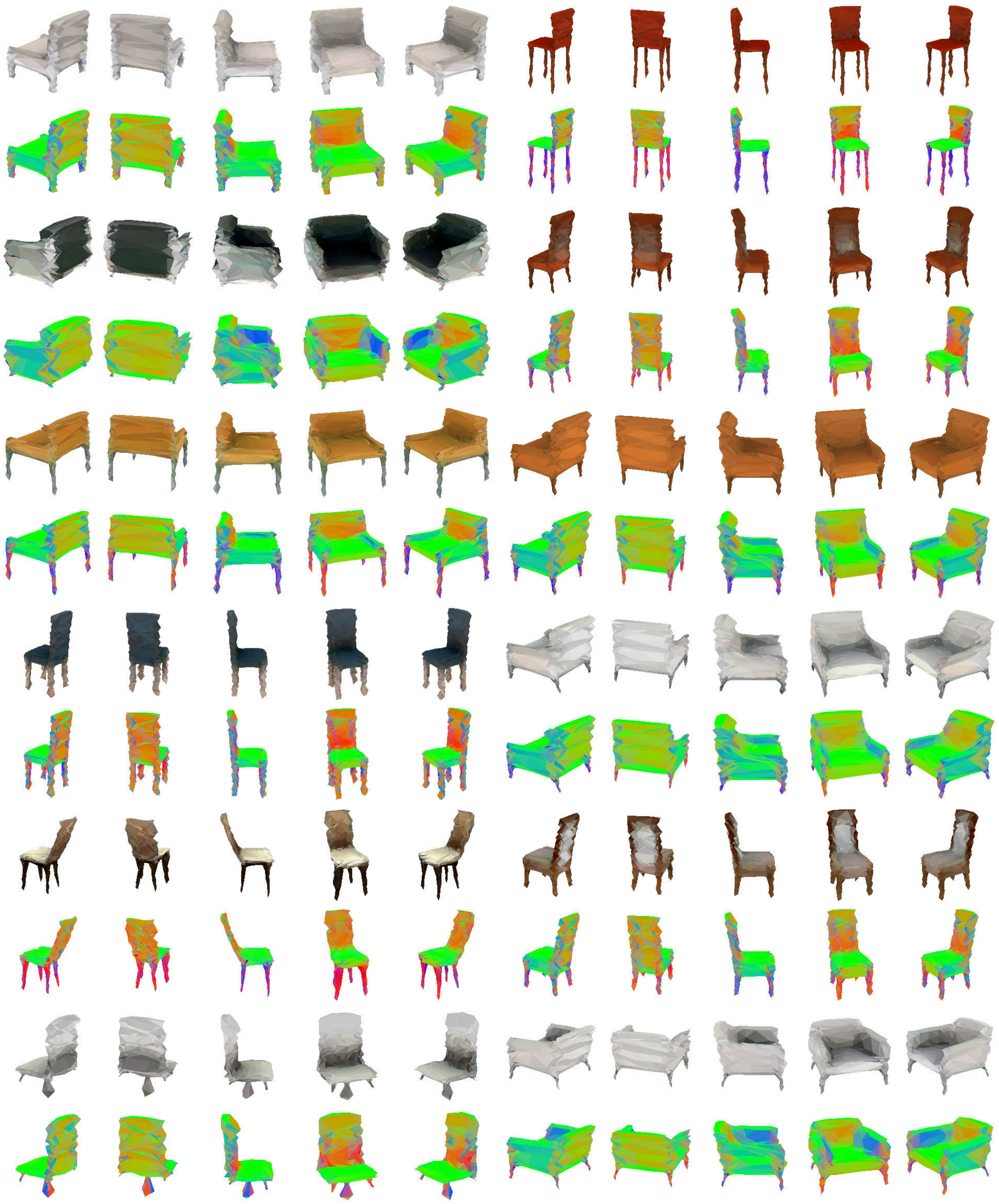

Figure S13: Examples of chairs generated by our model, in setting (NO-MASK) with parametrization (DENSE).

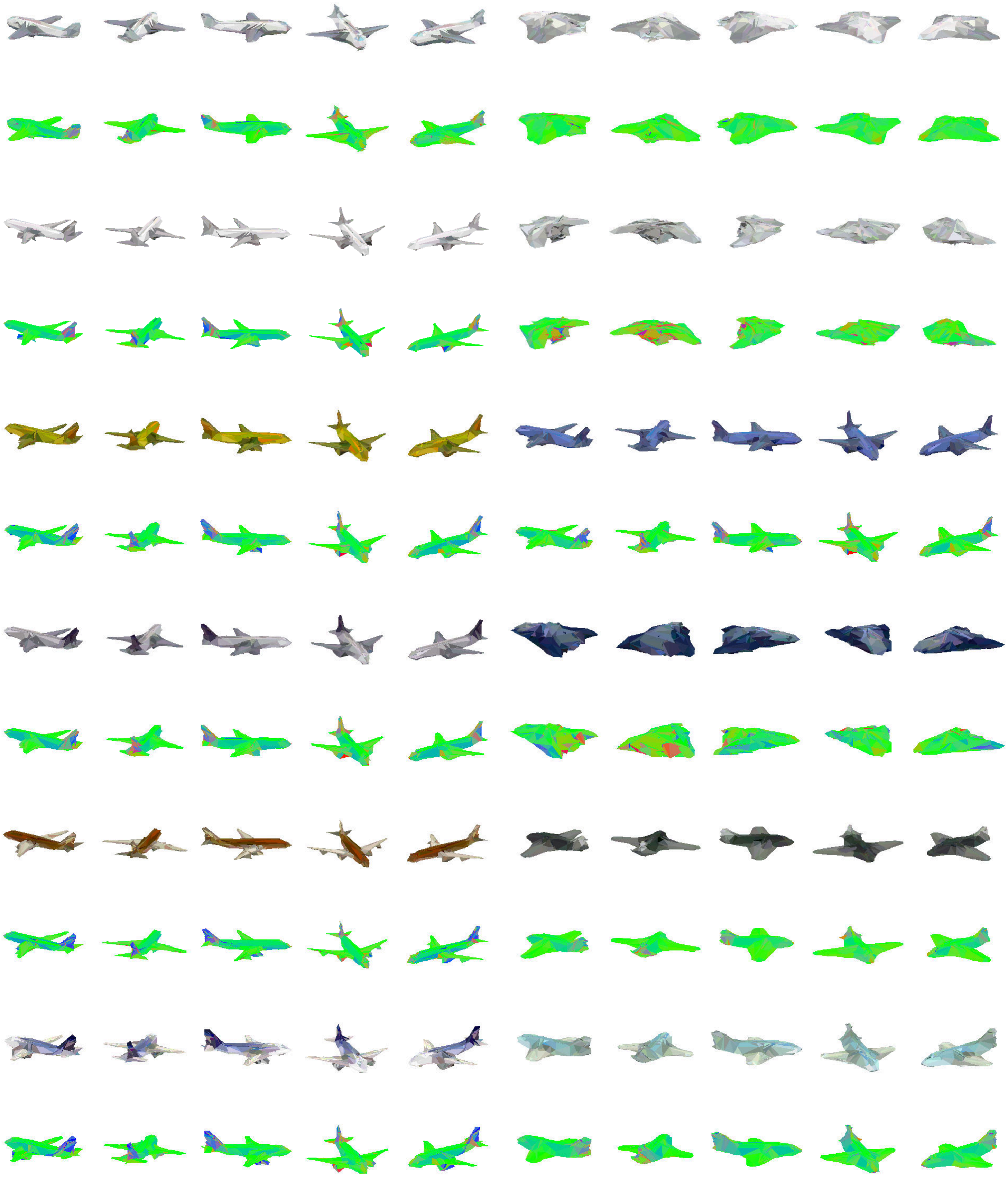
Figure S14: Examples of airplanes generated by our model, in setting (MASK) with parametrization (DENSE).

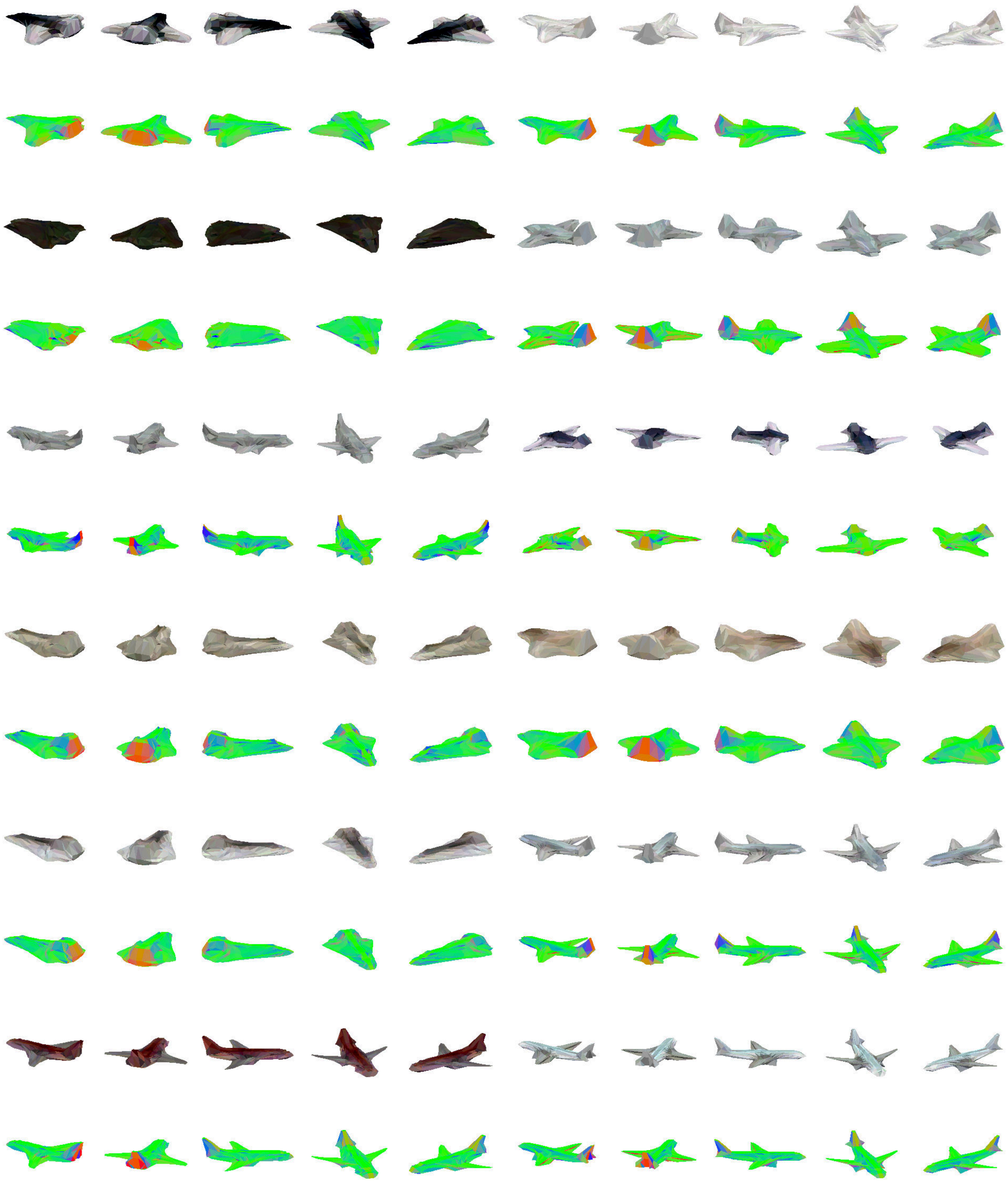

Figure S15: Examples of airplanes generated by our model, in setting (MASK) with parametrization (PUSHING).

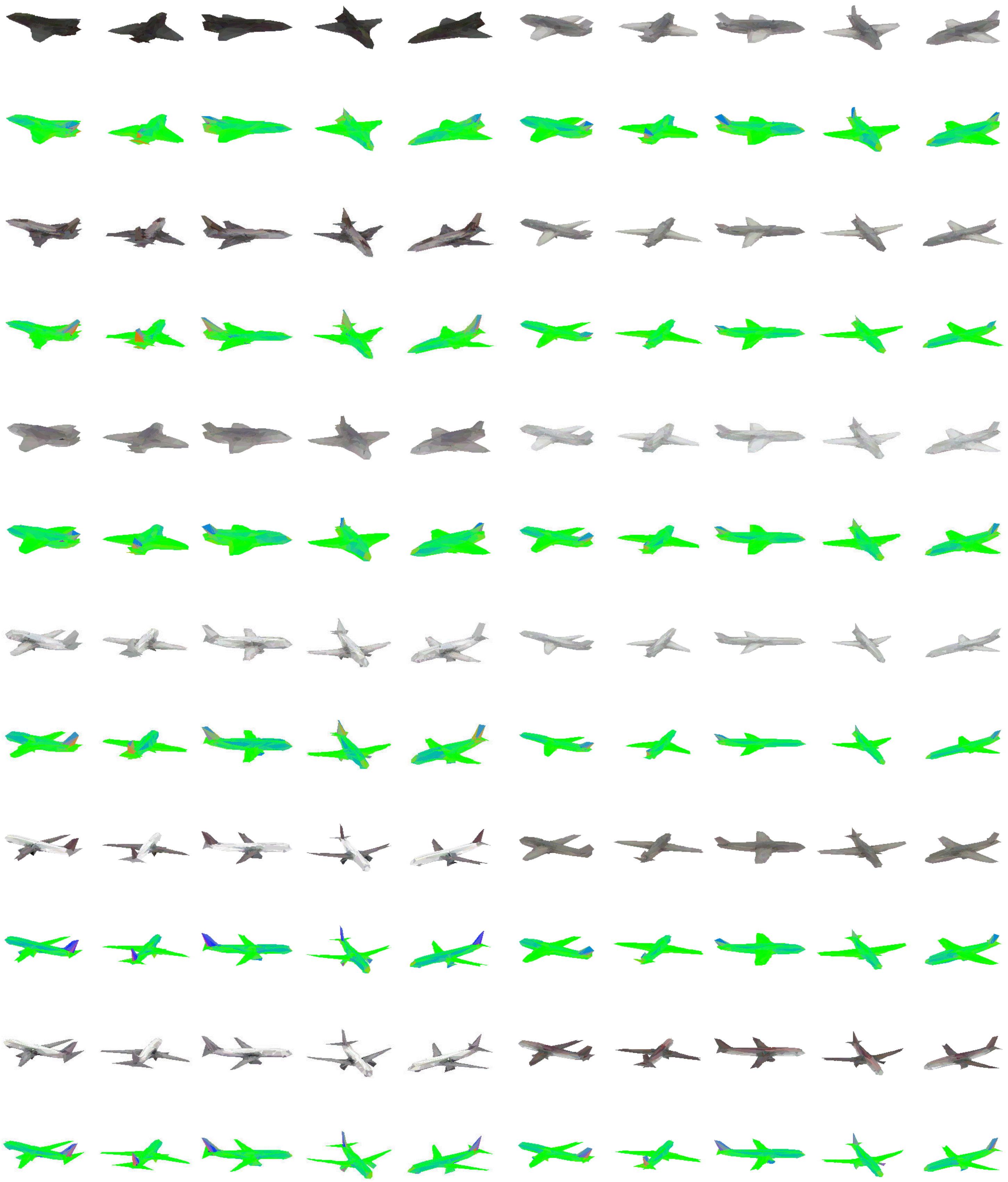

Figure S16: Examples of airplanes generated by our model, in setting (NO-MASK) with parametrization (DENSE).

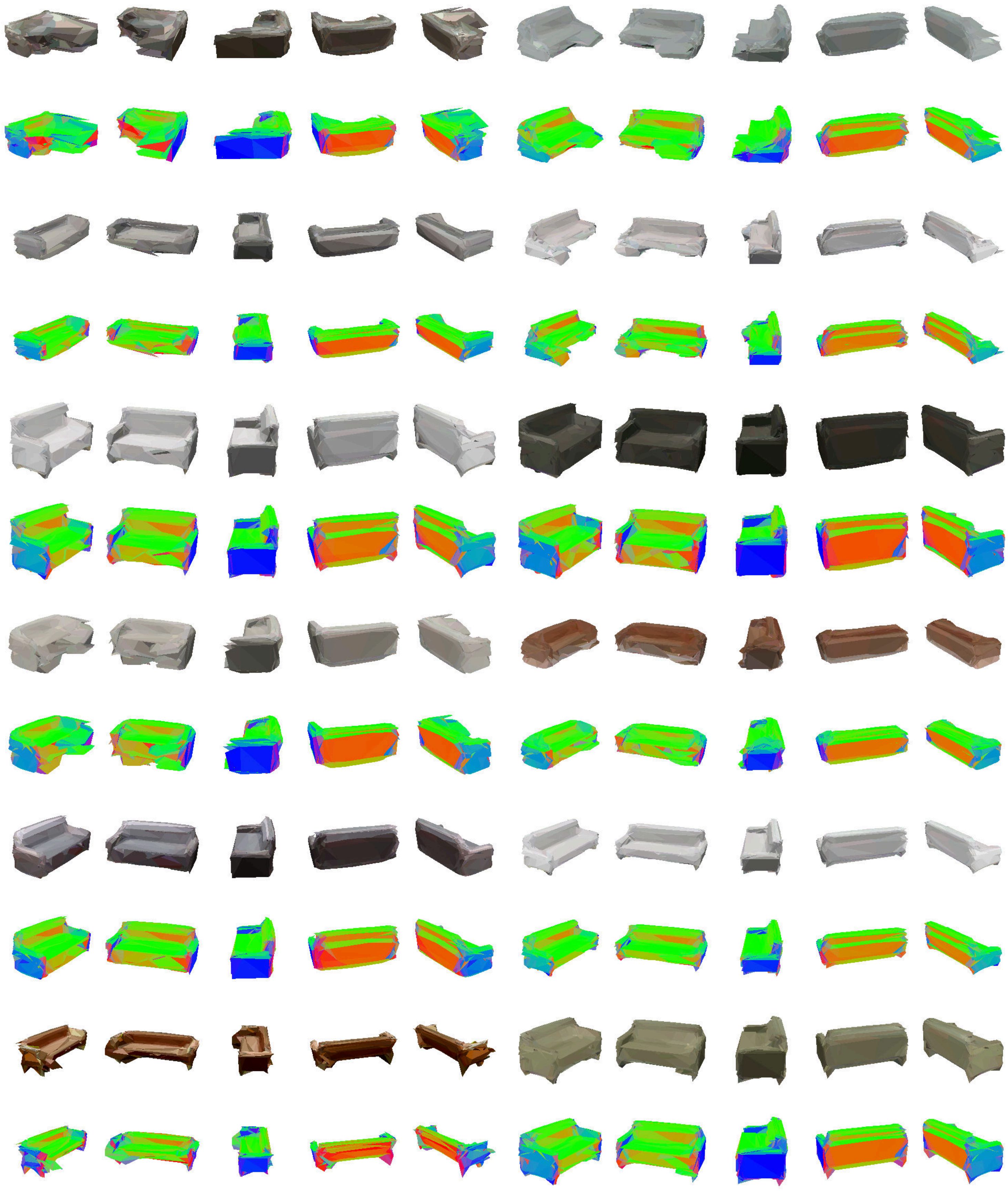
Figure S17: Examples of sofas generated by our model, in setting (MASK) with parametrization (DENSE).

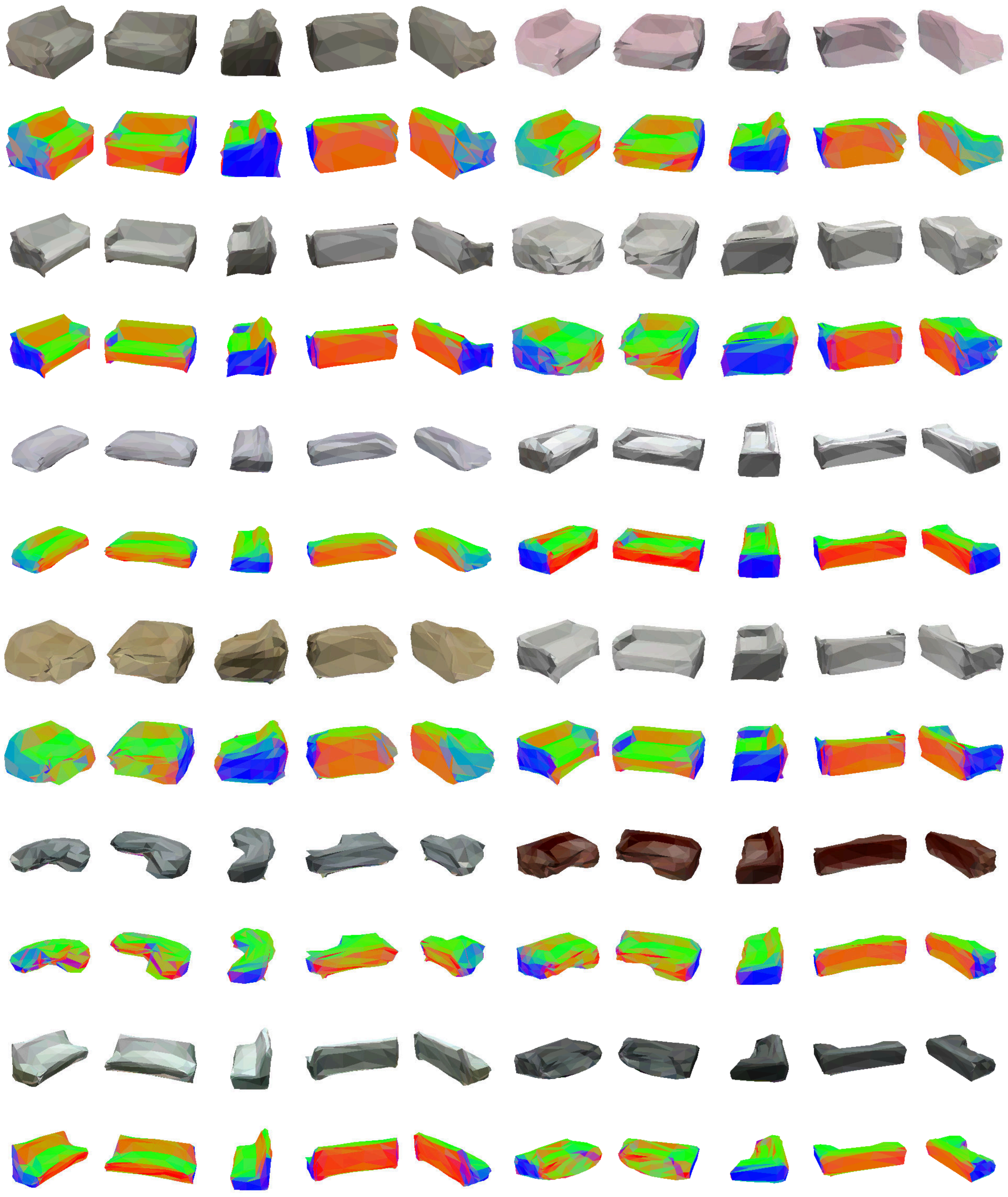

Figure S18: Examples of sofas generated by our model, in setting (MASK) with parametrization (PUSHING).

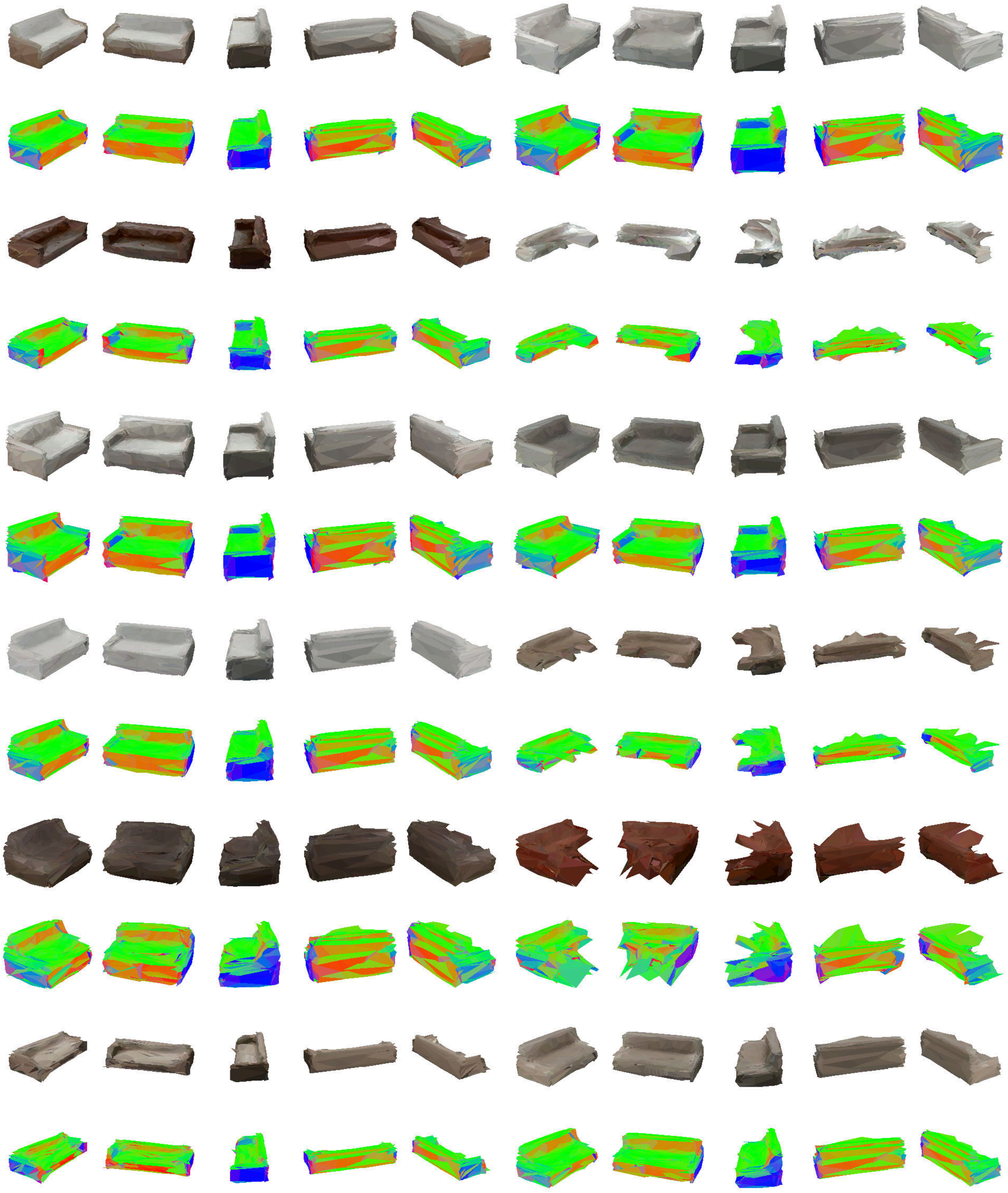
Figure S19: Examples of sofas generated by our model, in setting (NO-MASK) with parametrization (DENSE).