Volume xx (200y), Number z, pp. 1–4

A Generative Framework for Image-based Editing of Material Appearance using Perceptual Attributes

1. Additional Details on the Framework

Our framework is composed of two encoder-decoder networks $\mathcal{G}_1$ and $\mathcal{G}_2$, the auxiliary latent discriminator networks $\mathcal{LD}_1$, $\mathcal{LD}_2$ and the auxiliary attribute predictor and discriminator $\mathcal{C}/\mathcal{D}$ only used by means of a loss function during training.

Generative networks Both generative networks $\mathcal{G}_1$ and $\mathcal{G}_2$ are composed of an encoder made of a series of convolutional blocks that reduce the spatial dimensions of the input by a factor of two, a set of residual blocks that transform the bottleneck features, and a decoder made of a series of convolutional blocks followed by bilinear upsampling layers. The target perceptual attribute is spatially replicated to match the size of the latent code and concatenated to it at the beginning of the decoder.

Let `Ck` denote a 4×4 Convolution layer with k filters and stride 2, then followed by a Rectified Linear Unit (ReLU), `Rk` denotes a residual block that contains two 3×3 convolution with k filters. `Dk` denotes a convolutional block (3×3 convolution with k filters - leaky Rectified Linear Unit [XWCL15]) followed by a bilinear upsampling layer. Reflection padding is used in all convolutions.

$\mathcal{G}_1$ takes input images at the resolutions 128×128 and contains six layers both in the encoder and decoder and two residual blocks, resulting in the following architecture:

Encoder: `C32-C64-C128-C256-C512-C512-`
Bottleneck: `-R512-R512-`
Decoder: `-(b)D512-D256-(n)D128-(n)D64-(n)D32-(n)D8`

where `(b)` indicates the concatenation of the target attribute and `(n)` indicates the concatenation of the normal map.

$\mathcal{G}_2$ takes as input images at the resolution 256×256 and contains four layers in the encoder, three in the decoder and three residual blocks, resulting in the following architecture:

Encoder: `C32s1k7-C64-C128-C256-`
Bottleneck: `-R256-R256-R256-`
Decoder: `-(b)(n)D128-(n)D64-(n)D8`

where `C32s1k7` indicates a 7×7 Convolution-ReLU layer with 32 filters and stride 1. This first convolution allows us to reduce the number of spatial resolution of the image while keeping the same receptive field.

Each network ends with a last convolutional block with stride 1 and 8 filters followed by a single convolutional layer with three output filters (corresponding to the RGB channels) and a hyperbolic tangent function (*tanh*) to bring the values into the range $[-1,+1]$.

Latent discriminator The latent discriminators, $\mathcal{LD}_1$ and $\mathcal{LD}_2$ take the features in the bottleneck of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, and use them to predict the attribute a of the input image. The architecture of the latent discriminators $\mathcal{LD}_1$ is as follows:

$\mathcal{LD}_1$**:** `Cd512-FC256-FC1`
$\mathcal{LD}_2$**:** `Cd512-Cd512-Cd512-Cd512-pool-FC256-FC1`

where `Cdk` represent a convolutional block (4×4 convolution, leakyReLU, and dropout with probability 0.3), `FCk` refers to a fully connected layer with `k` features, and `pool` represent an average pooling operation. At the end, the output of the latent discriminators goes through a *tanh* layer that outputs the attribute prediction $\hat{a}$ in the range $[-1,+1]$.

Attribute predictor and discriminator The attribute predictor and discriminator $\mathcal{C}/\mathcal{D}$ take the image as input and outputs an attribute prediction $\hat{b}$. The image goes first through an encoder. The features from such encoder then go to the discriminator, and the attribute predictor. The architecture is as follows:

Encoder: `C32-C64-C128-C256-C512-`
Discriminator: `-C1`
Attribute predictor: `-pool-FC256-FC1`

WGAN-GP loss formulation Generative Adversarial Networks (GANs) are complex to train. This is partially due to the instability of the loss function proposed in the original formulation [GPAM*14]. WGAN-GP [GAA*17] aims to alleviate such problems by introducing a new loss function that relies on the Wasserstein distance between distributions and a gradient penalty term $\mathcal{L}_{GP}$.

Intuitively, the discriminator is trained to give a high score to real images and a low score to generated ones, aiming at disambiguate them:

$$\mathcal{L}_{adv}(\mathcal{D}) = -\|\mathcal{D}(\mathcal{I})\|_2 + \|\mathcal{D}(\mathcal{G}(\mathcal{I},n,b))\|_2 + \mathcal{L}_{GP} \quad (1)$$

while the generator is trained such that the the discriminator believe that generated images are actually real (giving them a high score):

$$\mathcal{L}_{adv}(\mathcal{G}) = -\|\mathcal{D}(\mathcal{G}(\mathcal{I},n,b))\|_2 \tag{2}$$

We refer the reader to the original manuscript for additional information [GAA*17].

Data augmentation To have a more diverse set of input images and help the model generalize better, we perform a set of random data augmentation routines. First, input images are scaled to have size 512×512 px and we perform random flips, 90-degree rotations, and a random crop with size 480×480 px. Then, to account for the bias in the BRDFs from the training dataset, we perform random changes in the saturation and the hue. Finally, the image is scaled to 256×256 and fed to the networks.

2. Additional Details on the Normal Prediction

Our normal map prediction module uses as input single-views of RGBA images. The architecture is based on the Pix2Pix network [IZZE17], which has been shown to perform reasonably well in normal prediction tasks [SSSJ20, NSH*19, GFM*19]. Our goal is to maintain as much geometrical detail as possible, while making the normal predictions invariant to changes in material and illumination conditions in the input images.

2.1. Architecture

Our network takes RGBA images as input (RGB + background mask), and follows an encoder-decoder architecture, with 4 downsampling blocks in the encoder and 4 upsampling blocks in the decoder. In each block we repeat twice the following structure: Convolution with kernel 4×4, a batch-normalization layer, and a leakyReLU [XWCL15]. This is done in order to reduce the impact of specular reflections in the final predictions, putting more space between the skip connection and the final output of the network. We also included residual connections within each block, as proposed by ResNet [HZRS16]. Residual connections stabilizes the network and reduces the amount of high variance noise present in the predictions. In contrast to Pix2Pix, which uses transposed convolutions, we use bilinear upsampling in order to reduce the risk of checkerboard artifacts. The final architecture is the following one:

Encoder: `R64-ER64-ER128-ER256-ER512-`
Bottleneck: `-R512-`
Decoder: `-DR512-DR256-DR128-DR64-R64`

where `ER` indicates an encoder block (downsampler) with residual connections, `DR` a decoder block (upsampler) with residual connections, and `R` a convolutional block with residual connections. The number that follows them indicates the number of filters used in the convolutions. The output uses a hyperbolic tangent function (*tanh*), bounding the results of the predictions to $[-1,1]$, which are then scaled to have unit length, and normalized to the range $[0,1]$. The network's weights are initialized with a zero-mean normal distribution and a standard deviation of 0.02.

2.2. Losses

Our loss function is described in Equation 3 and it is composed of three different losses: an adversarial loss $\mathcal{L}_{adv}$, a perceptual loss $\mathcal{L}_{vgg}$, and a reconstruction loss $\mathcal{L}_{rec}$.

Adversarial loss To infer normal maps similar to their groundtruth distribution we rely on an adversarial loss $\mathcal{L}_{adv}$ with a binary cross entropy (BCE) function. We rely on the same discriminator model as the one proposed in Pix2Pix [IZZE17].

Perceptual loss To keep high-frequency details in the inferred normals we include a perceptual loss [JAFF16] $\mathcal{L}_{vgg}$. To extract image features we employ the VGG16 [SZ15] model pretrained on ImageNet [DDS*09] and compute feature differences with an L_1 loss.

Reconstruction loss To directly supervise the prediction of each normal we rely on a Mean Squared Error (MSE) function $\mathcal{L}_{rec}$. Since normal vectors have unit-norm, the MSE is equivalent to a cosine distance, which has additional geometric properties.

To obtain our final loss we set the different weights to $\lambda_{adv} = 0.25$, $\lambda_{rec} = 10$, and $\lambda_{vgg} = 1$. Our final loss function is:

$$\mathcal{L} = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{vgg}\mathcal{L}_{vgg}. \tag{3}$$

2.3. Training

The model was trained on synthetic data with paired ground-truth normal maps. The synthetic dataset was composed of 12 different geometries, with 5 different viewpoints, 6 different illumination conditions, and 100 different materials each; accounting for a total of 42000 images of size 128×128 px. We implemented several data augmentation techniques, including random 90 degree rotations, flips, and random gamma, hue, saturation, and brightness changes. Adam optimizer [KB14] is used with an initial learning rate of 0.0007, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Our network is implemented using Pytorch [PGM*19] and Pytorch Lightning [Fal19] as our frameworks. The model was trained until evaluation losses plateaued for more than 10 epochs, which usually occurred after around 70 epochs. Overall, training took 7 hours in a single NVIDIA RTX 3080 and an AMD Ryzen 9 5900x.

3. Additional Details on the Perceptual Study

Figure 1 shows a screenshot of the perceptual study, as seen by the participants. The stimuli is shown on the left part of the screen while the list of attributes to score are shown on the right.

Training of participants Participants of our perceptual study first had to go through a training session in which they were shown with a text description and a few example images depicting materials with low and high score values for each attribute. We then show them the same screen as in the study and ask them to answer the attributes for two easy examples (shown in Figure 2, left). If answers of the participants were not the expected ones, we instructed them to look again at the image and check the description of the attributes.

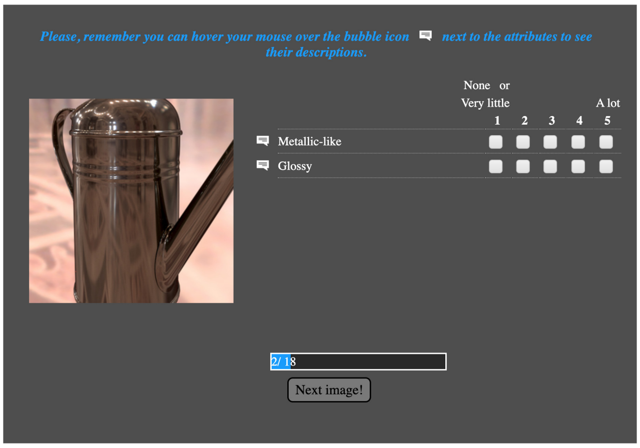

Figure 1: *Screenshot of the perceptual study as seen by participants. Stimuli is shown on the left, the participant have to select a score for the two attributes shown on the right.*

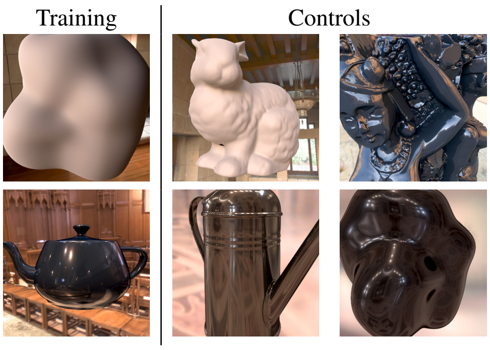

Figure 2: *Left: the two images used in our training session. Right: the four images used as controls.*

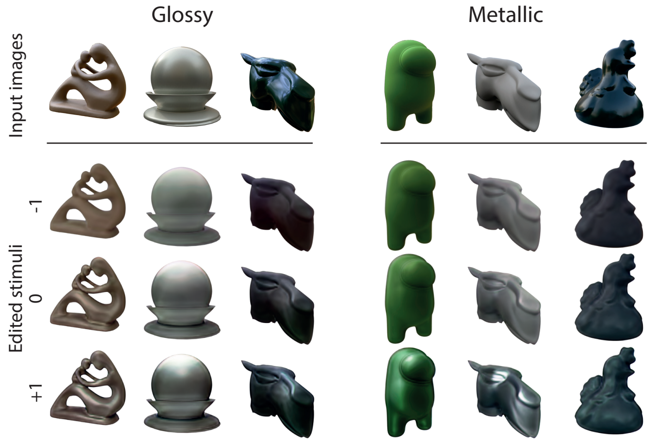

Figure 3: *Input images and edited stimuli used in our user-study. Top: input images to our framework. Bottom: The edited images with three target attributes, leading to nine stimuli for each attribute.*

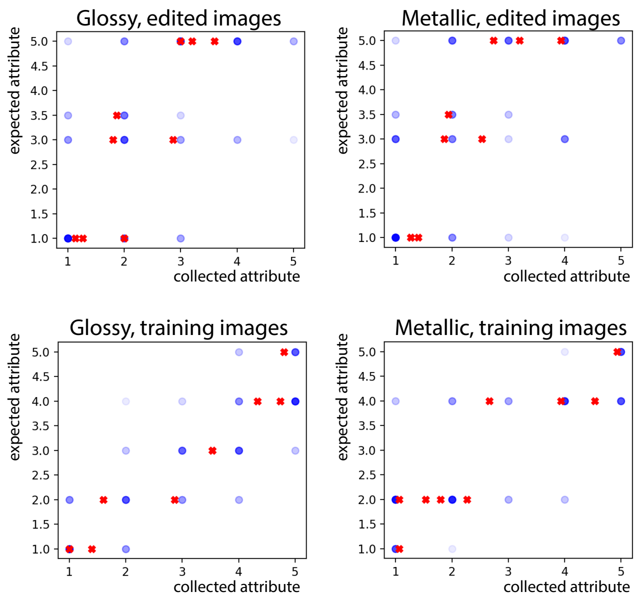

Figure 4: *Answers collected in our validation study for both attribute Metallic and Glossy and for the two sets of images. The blue dots show all the 15 ratings that we collected for each images, where the density of the color indicates the number of answer, while the red crosses indicates the average answer for each stimuli.*

Control Questions In addition to the 15 stimuli, we added four control images in order to detect lazy users. These images contains materials with clear expected answers (shown in Figure 2 right). We rejected participants answering wrongly to more than one of these questions and rejected 20% of the participants based on this criteria.

4. Additional Details on the Validation Study

The layout of the user study is the same as the one used in the perceptual study (Figure 1) except that participants were asked to rate one attribute at a time.

In Figure 3, we show the stimuli from the "edited images" set that we used in the validation user study. For each attribute, the top part shows the input images (synthetic) that we selected, covering different shapes, illuminations and reflectance properties. The bottom part shows the three edited images that we show in the study for each input (low attribute value, middle value and high attribute value), resulting in nine stimuli.

In Figure 4, we show the answers that we collected for both attribute *Metallic* and *Glossy* and for the two sets of images. The blue dots show all the 15 ratings that we collected for each images, where the density of the color indicates the number of answer, while the red crosses indicates the average answer for each stimuli.

While the answers for both sets of images appear to be strongly correlated, the answers collected on our edited images do not reach the full scale of the attribute, with a maximum score of 3.7 for the

Glossy attribute, and 4 for the *Metallic* attribute. The average variances in the answers was higher for edited images than for training ones (0.42 and 0.62 respectively for *Glossy*, 0.5 and 0.84 respectively for *Metallic*).

5. Full Results of the Quality User Study

Figure 5 shows the answers collected in the quality user study for all the stimuli.

6. Additional Results

In Figure 6 and 7, we show results when editing real or synthetic images with the attribute *Metallic* and *Glossy* respectively, sampling the attributes at different values along their range.

References

[DDS*09] DENG J., DONG W., SOCHER R., LI L.-J., LI K., FEI-FEI L.: Imagenet: A large-scale hierarchical image database. In *Proc. Computer Vision and Pattern Recognition* (2009), pp. 248–255. 2

[Fal19] FALCON WA E. A.: Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* (2019). 2

[GAA*17] GULRAJANI I., AHMED F., ARJOVSKY M., DUMOULIN V., COURVILLE A.: Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* (2017). 1, 2

[GFM*19] GABEUR V., FRANCO J.-S., MARTIN X., SCHMID C., ROGEZ G.: Moulding humans: Non-parametric 3d human shape estimation from single images. In *Proc. International Conference on Computer Vision* (2019). 2

[GPAM*14] GOODFELLOW I. J., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014). 1

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *Proc. Computer Vision and Pattern Recognition* (2016). 2

[IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A. A.: Image-to-image translation with conditional adversarial networks. In *Proc. Computer Vision and Pattern Recognition* (July 2017). 2

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *Proc. European Conference on Computer Vision* (2016), Springer, pp. 694–711. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 2

[NSH*19] NATSUME R., SAITO S., HUANG Z., CHEN W., MA C., LI H., MORISHIMA S.: Siclope: Silhouette-based clothed people. In *Proc. Computer Vision and Pattern Recognition* (June 2019). 2

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., ET AL.: Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019). 2

[SSSJ20] SAITO S., SIMON T., SARAGIH J., JOO H.: Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proc. Computer Vision and Pattern Recognition* (2020). 2

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *CoRR* (2015). 2

[XWCL15] XU B., WANG N., CHEN T., LI M.: Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015). 1, 2

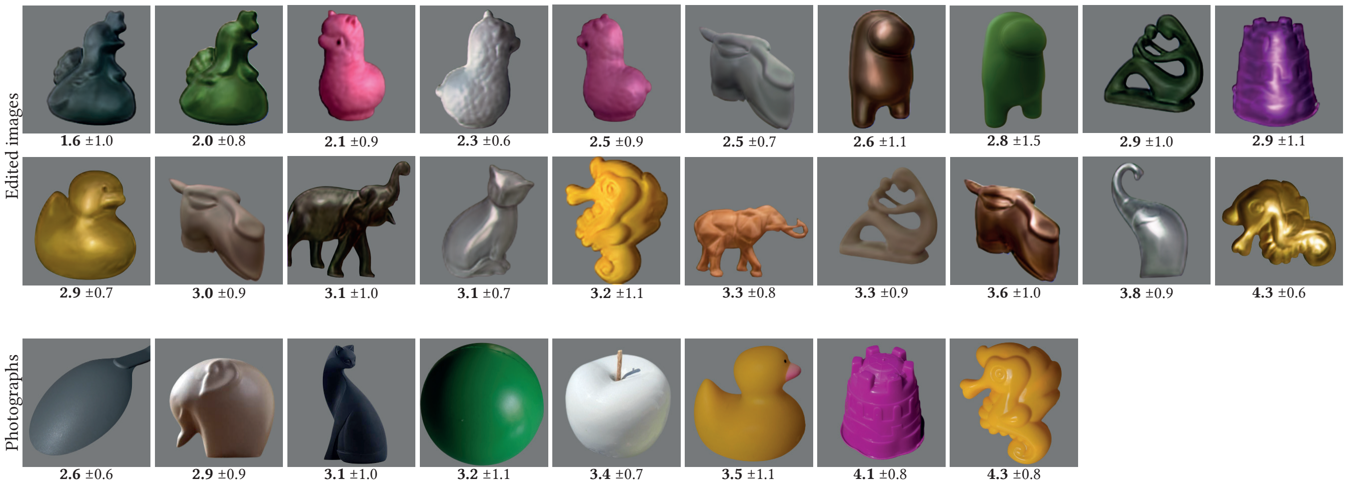

Figure 5: *Average scores of quality collected in the user study over the 20 edited images (top) and the 8 real photographs (bottom). We show under each image the average scores along with their standard deviation.*

Figure 6: *Editing results by varying the perceptual attributes* Metallic. *First column is the input image, following ones show the edited image when sampling the attribute as* $[-1,\ 0,\ 0.25,\ 0.5,\ 1]$.

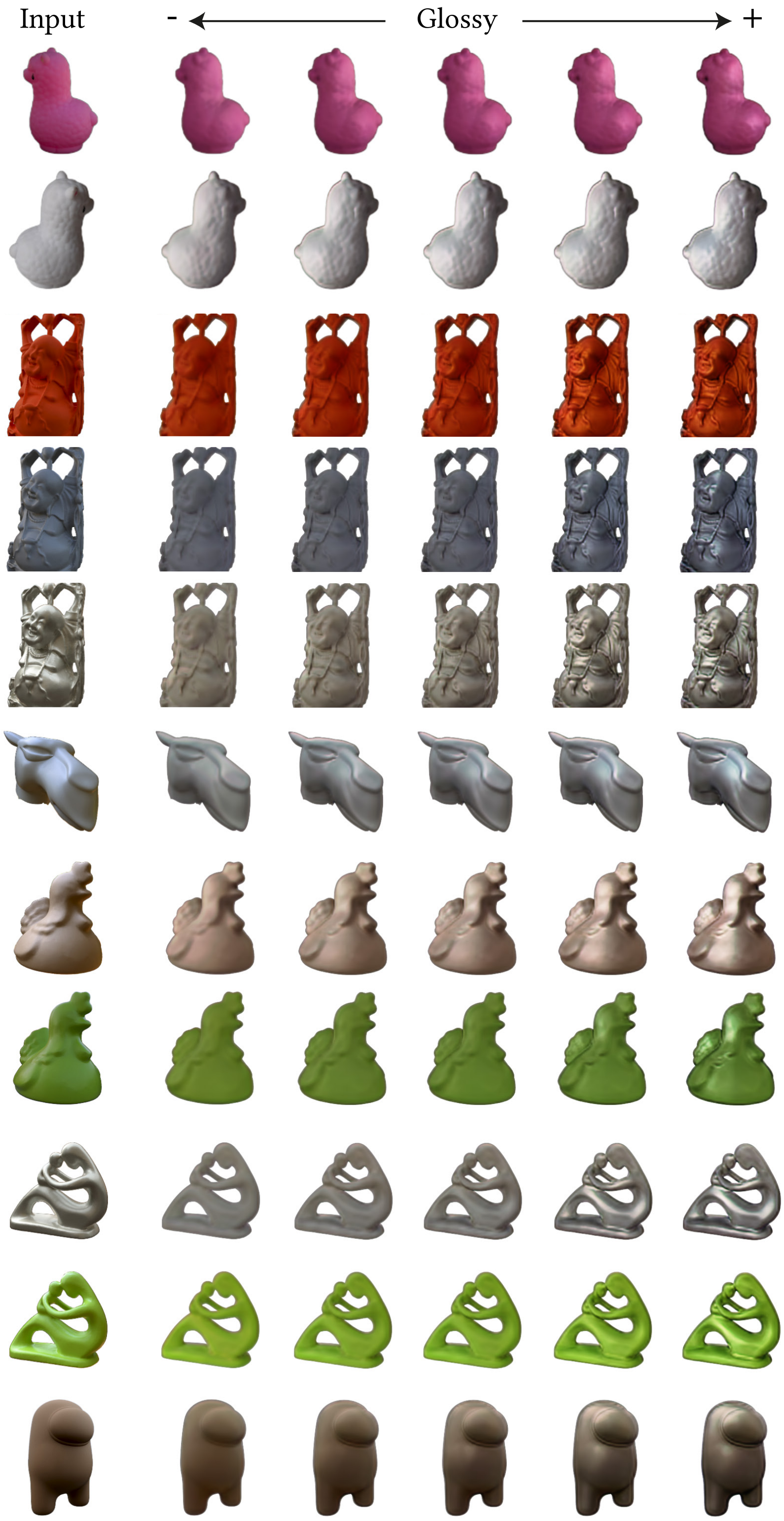

Figure 7: *Editing results by varying the perceptual attributes* Glossy. *First column is the input image, following ones show the edited image when sampling the attribute as* $[-1,\ -0.25,\ 0,\ 0.5,\ 1]$.