

## Controlling 3D objects in 2D image synthesis

Stephan Brehm, Florian Barthel, Rainer Lienhart

### Angaben zur Veröffentlichung / Publication details:

Brehm, Stephan, Florian Barthel, and Rainer Lienhart. 2022. "Controlling 3D objects in 2D image synthesis." SN Computer Science 4 (1): 48.  
<https://doi.org/10.1007/s42979-022-01462-w>.





# Controlling 3D Objects in 2D Image Synthesis

Stephan Brehm<sup>1</sup> · Florian Barthel<sup>1</sup> · Rainer Lienhart<sup>1</sup>

Received: 10 May 2022 / Accepted: 17 October 2022  
© The Author(s) 2022

## Abstract

In this work, we propose a method that enforces explicit control over various attributes during the image generation process in a generative adversarial net. We propose a semi-supervised learning procedure that allows us to use a quantized approximation of object orientation for learning continuous object rotations. As a result, among many other attributes, our proposed method allows us to control object orientation in scenes that are rendered according to our specifications.

**Keywords** Image synthesis · 3D rendering · Generative adversarial nets · Conditional · Regression · Rotation · Semi-supervision · Cooperation

## Introduction

Computer Graphics traditionally require a three-dimensional model of the objects that are to be rendered. These 3D models are then texturized and placed in a virtual three-dimensional space. This three-dimensional space is then projected to two dimensions in order to show the result on a screen. In this work, we explore the possibility of embedding three-dimensional knowledge in a deep neural net. We propose to omit 3D modeling and subsequent 2D projection by directly rendering the 2D result from a given specification of the desired scene. Adherence of the resulting deep neural renderer to this specification is key to obtain useful results. A particular focus of this work is to achieve fine-grained control on the orientation of objects in the resulting scenes.

Our contributions are as follows:

1. We propose a training procedure that explicitly enforces the adherence of a generative adversarial model to a given specification of the scene that is to be rendered.
2. We propose a method to learn continuous rotations from data that is only annotated categorically, i.e., angles are quantized to the nearest 45°. We propose to leverage the continuity and periodicity of rotations and extend the formulation of conditional GANs to such continuous targets. This effectively results in a regression-based GAN formulation.
3. The above-mentioned categorical annotations of rotations induce label sparsity when reinterpreted as continuous labels, i.e., our data do not provide any examples for angles in between the quantized annotations. To overcome this limitation, we propose a semi-supervised method that forces generator and discriminator to find agreement for these intermediate rotations.
4. Given the novel focus on generation of continuously changing object orientations, we propose novel methods for evaluation.
5. We present a dataset that is annotated with a huge variety of attributes.

This work is organized as follows. In "[Related Work](#)", we first discuss relevant related work. In "[Dataset](#)", we introduce our dataset including the annotated attributes which in combination form possible specifications. In "[Goals and Metrics](#)", we discuss and propose metrics for evaluating our proposed system. In "[Conditional Object Synthesis](#)", we introduce our approach to enforce adherence of our model to the specifications and discuss the proposed method for learning continuous rotations. In "[Results](#)", we analyze our

---

✉ Stephan Brehm  
stephan.brehm@uni-a.de  
Florian Barthel  
florian.barthel@uni-a.de  
Rainer Lienhart  
rainer.lienhart@uni-a.de

<sup>1</sup> Department of Computer Science, University of Augsburg, Universitätsstraße 6, 86159 Augsburg, Bavaria, Germany

results both qualitatively as well as quantitatively. "**Conclusion**" concludes this work.

## Related Work

In this section, we cover related work for controlling the output of synthesis methods and related work for generative adversarial net (GAN)-based 3D rendering. Afterwards, we cover the StyleGAN2 [1] method for image synthesis which is the basis for our method.

Control in Image Synthesis Methods In *GANSpace: Discovering Interpretable GAN Controls* [2], Härkönen et al. propose a method for analyzing the latent space in GAN-based methods. They identify important latent directions based on Principal Components Analysis (PCA) applied in activation space [2]. From PCA in activation space, they derive directions in the latent space that correspond to a change in certain semantics. In StyleGAN-based networks, they show that they can find combinations of layer-wise changes that allow to target certain semantics of the generated images. In contrast to that, our method enables direct control over important semantics because we use semantic labels to directly control the learning of our model.

*GAN-based 3D rendering* GANs can generally be considered neural renderers. The result of this process typically is a 2D image in RGB-space or similar domains [3–5] Lately, methods for GAN-based 3D rendering have been proposed. They typically aim to construct a 3D model of an object or the scene [6–8]. In *Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering* [7], Zhang et al. propose a model that uses an inverse graphics network, i.e., a network that predicts 3D meshes, lighting and textures from 2D images. They use this information to subsequently render a 3D scene with a differentiable renderer. To learn this network, they use a StyleGAN as a *neural renderer* to create images with approximate orientations. To do this, they first needed to find the parts of the latent code as well as the important layers that control these orientation manually. With these synthesized images, they then train the inverse graphics network. In contrast to Zhang et al., we do not do inverse graphics but rather aim for the opposite, i.e., we aim to imitate 3D rendering with subsequent projection to two dimensions without ever requiring a 3D representation explicitly. However, we aim to explicitly control object rotation which is a three-dimensional property. In *GAN-Control: Explicitly Controllable GANs* [9], Shoshan et al. present a model that is capable of controlling image semantics directly. They propose to enhance adversarial training with additional contrastive objectives that they learn for each controllable attribute. Given the resulting generative model, they then learn an encoder that estimates a suitable latent code from a given specification. They also showcase limited control over the orientation of

generated objects, i.e., in contrast to ours, their model cannot perform full rotation cycles.

## StyleGAN2

The *StyleGAN2* model [1] is the basis for this work. It is a deep generative model for image synthesis that has gained a lot of popularity due to its capability of generating highly realistic images. As the name implies, this model is learned in an adversarial setting, i.e., the learning procedure requires a discriminative model that is used to supervise the generative model. However, the generative model differs greatly from standard convolutional feed-forward architectures. This is due to the explicit separation of the convolutional image generation from something that the authors term as *style*. Here, *style* refers to the relative importance of individual learned features that are used to compose the resulting image.

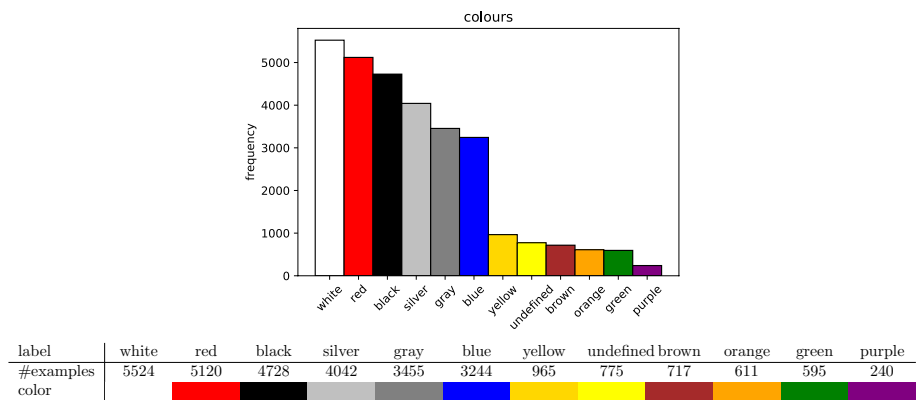
## Architecture

The StyleGAN2 architecture consists of two feed-forward networks running in parallel. For this work, we will term them as the *dense* stream and the *convolutional* stream. The convolutional stream is the part of the model that performs the actual image generation task, i.e., it takes an input  $c_1$  and transforms it into an image by multiple convolution and up-scaling operations. However, the input  $c_1$  to this main convolutional stream is constant. Thus, the convolutional stream, by itself, would only ever produce a constant output. Variance in the output is produced by the dense stream. This dense stream first embeds a random latent input  $z \in Z$  in a mapped latent space  $W$ . Embeddings  $w \in W$  are then used to modulate the weights of the main convolutional stream, i.e., individual filters of the convolution layers are scaled in order to reduce or increase their relative importance. Note that,  $Z$  does not need to be completely random. In fact, we can use the dense stream to explicitly control the output of the convolutional stream by extending individual latents  $z$  with conditioning information, i.e., add information about the desired image to the latents. For this work, we will utilize an abundance of attributes like object color, orientation, and many more, to precisely control the output of the generative model. Learning such attributes requires a dataset with corresponding annotations which we will further detail in "**Dataset**".

## Dataset

For this work, we use an extended version of the *cars30k* dataset [10]. This extended version contains a total of 73,784 images with multiple new annotations. In addition to the

**Fig. 1** Histogram of car body colors in our dataset sorted by frequency of the individual colors



existing body color annotations, we annotated body style, background scenery, and the rotation of the cars. To speed up the annotation process we adopted the following strategy:

- We annotate approximately 600 images for every new class. We find these images by selecting a few images manually and searching for nearest neighbors via a perceptual distance, i.e., we are looking for nearest neighbors in the deep features of a convolutional neural network (CNN). We then select matches manually.
- We keep  $\approx 120$  images per class for testing and use the remaining images to finetune a Resnet50 [11] that is pre-trained on the ImageNet dataset [12]. These classifiers typically achieve  $\approx 90\%$  accuracy on the test data.
- We use the learned classifiers to label the remaining images automatically.
- We check the resulting classifications manually and remove all images that are not correctly assigned to each of the new classes. Thus, we can safely assume that most of our annotations are correct, i.e., annotation accuracy is considerably higher than our 90% test accuracy. Note that the removed images not only include wrong labels but also include some rare instances of images that show multiple cars or images of the interior of cars.

**Body Colors**

In Fig. 1, we show the distribution of body colors. We distinguish between 11 different colors as well as an additional undefined/other color which describes all types of cars that are not colored uniformly. In total, we have 30,016 images that are annotated with the *body colors* of the cars that are shown.

**Manufacturers and Models**

Figure 2 visualizes the distributions of car manufacturers and car models in the dataset. There are 18 different manufacturers and 67 different car models annotated in the

dataset. In total, we annotated 70.461 images with *manufacturer* labels and 70.420 images with *car model* labels.

**Object Rotation**

In Fig. 3, we illustrate our rotation annotations. Note that the crafting of precise object rotation annotations is virtually impossible as it involves manual estimation of 3D object pose. Such a process would not only require precise manual annotations but also requires the camera to be calibrated. Both requirements are infeasible or even impossible to fulfill when working with images that are randomly downloaded from the internet. Thus, we opt for a simpler annotation process. Similar to the color annotations we adopt a quantization process. We quantize the rotations of cars shown in the images to eight categorical rotations (see Fig. 3 for a listing). This categorical definition of object rotation simplifies the annotation process massively because it does not require to measure the exact rotation of an object. As visualized in Fig. 3, we define the rotation as the direction of the hood of the car as seen from the position of the camera. Instead of measuring, we can annotate the rotation by simply identifying the approximate direction of the hood of a car, e.g., the annotated class is *profile left* if the car is visible from the side and the hood points to the left or similarly the annotated class is *front center* if the hood of the car points directly towards the camera.

**Body Style**

In Fig. 4, we illustrate all different *body style* categories that we annotated in our dataset. Note that the hatchbacks form the most frequent class while off-road vehicles and pickup trucks are not as common in the dataset. As such, the distribution of body styles probably resembles a distribution that is closer to a European distribution of body styles. In total, 39,187 of all 73,784 images are annotated with the *body style* of the cars that are shown.

**Background Scenery**

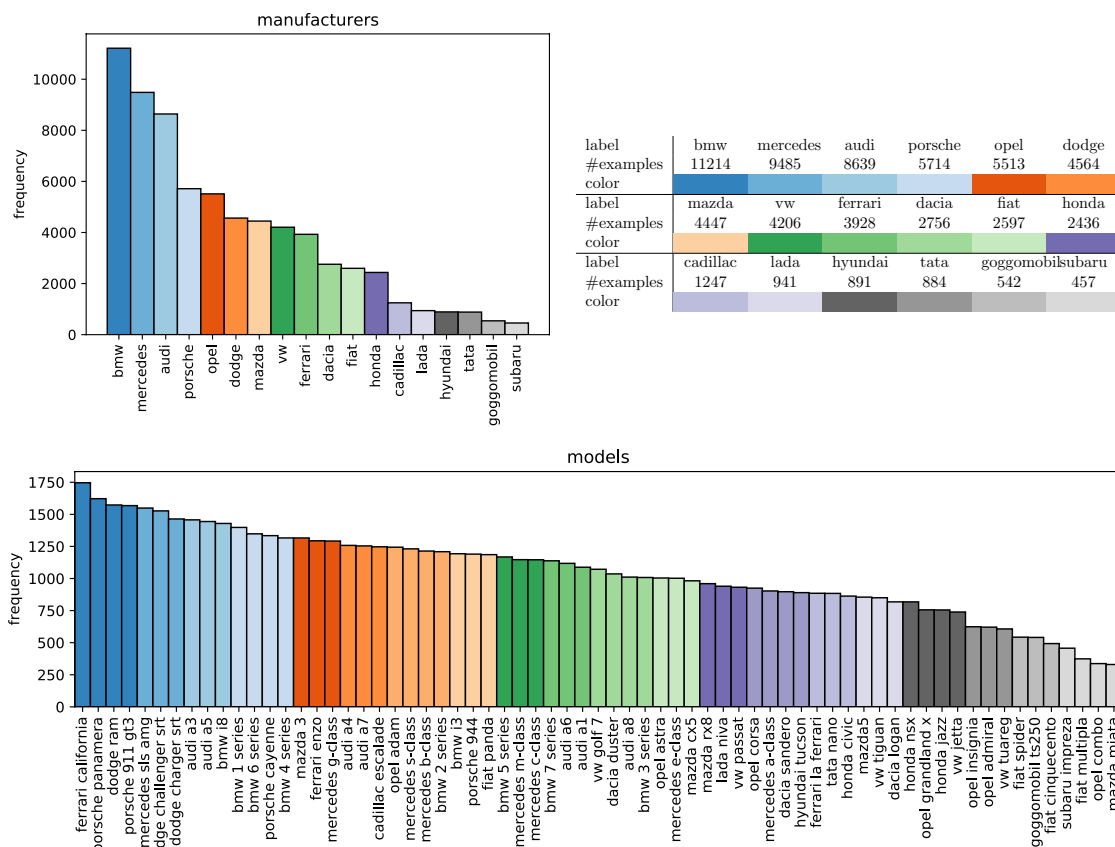


Fig. 2 Distribution of car manufacturers and car models in the dataset. Top: car manufacturers. Bottom: car models

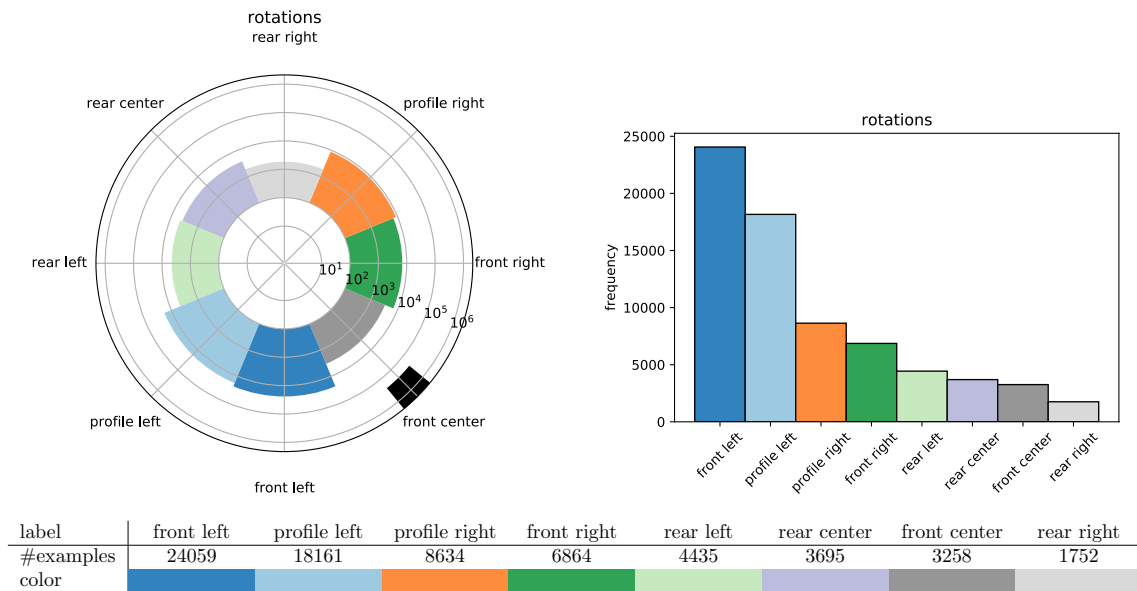
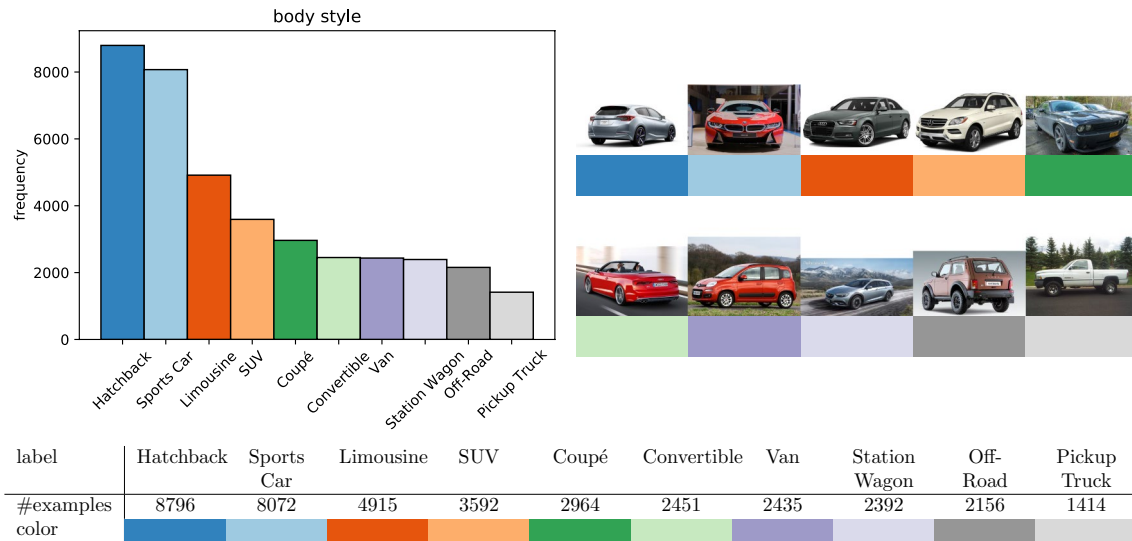
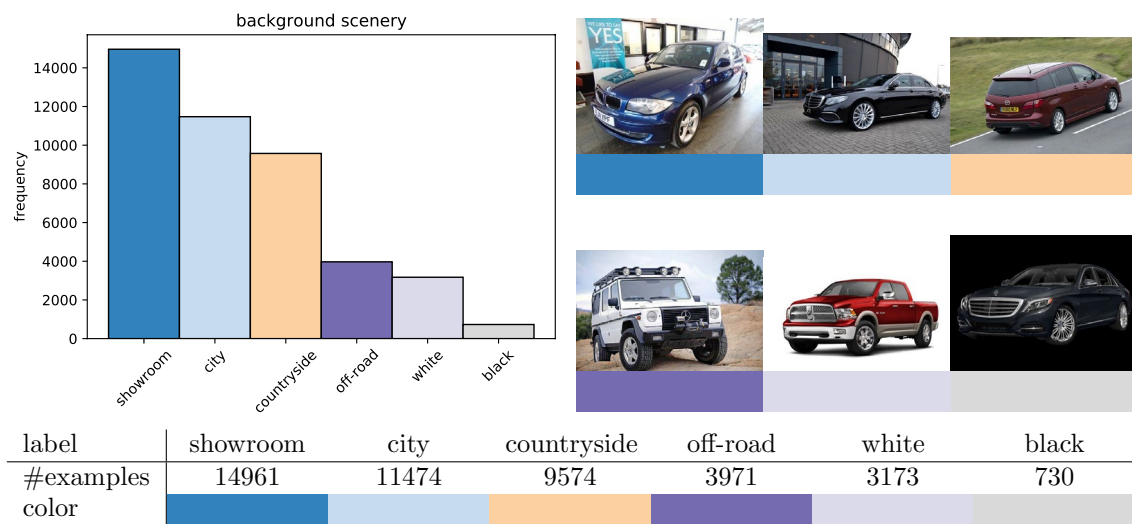


Fig. 3 Different rotations labeled in our dataset. We quantize rotations to 8 discrete labels. Top: histograms of rotation frequency. Top left: histogram (logarithmic scale) sorted by object rotation in a top-down view. The black box represents the camera. Rotation

labels describe the orientation of a car, i.e., the direction in which the hood of the car points as observed from the camera. Top right: histogram (linear scale) sorted by the frequency of the individual rotations. Bottom: table of class frequencies and legend



**Fig. 4** *Body style* labels in our dataset. Top left: histogram of *body style* labels. Top right: color-coded examples for each of the annotated body styles. Bottom: Table of class frequencies and legend



**Fig. 5** *Background* labels in our dataset. Top left: histogram of *background* labels. Top right: color-coded examples for each of the background labels. Bottom: table of class frequencies and legend

In Fig. 5, we illustrate the different background annotations in our dataset. We differentiate six different types of common background sceneries:

1. *Showroom* Mostly indoors, often some sort of marketing poster in the background.
2. *City* Outdoors, usually on a street, buildings in the background.
3. *Countryside* Outdoors, usually on a street, nature in the background.
4. *Off-road* Outdoors, no street, nature in the background.

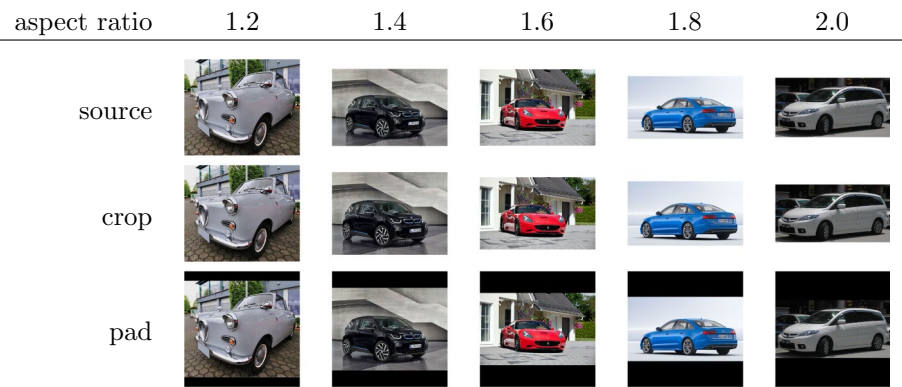
5. *White* No scenery, white background.
6. *Black* No scenery, black background.

In total, 43,883 of all 73,784 images are annotated with one of the above *background scenery* labels.

**Aspect Ratio**

We quantize aspect ratios of our images to five categories. For this purpose, we define the aspect ratio  $R_I$  of an image  $I$  in our dataset  $D$  as the ratio between image width  $W_I$  and image height  $H_I$ :

**Fig. 6** The aspect ratio quantization process. We categorize each image into the nearest of one of 5 aspect ratios. We then crop the image to the exact ratio. The resulting images are padded to a quadratic shape for training



$$R_I = \frac{W_I}{H_I}. \quad (1)$$

We quantize  $R_I \forall I \in D$  to the nearest category  $y_I^R \in [1.2, 1.4, 1.6, 1.8, 2.0]$ . Note that these are all aspect ratios for landscape images as portrait images are practically non-existent in the dataset. Note that we need these annotations to explicitly control the aspect ratios of the images that are generated by our proposed method. Due to this demand, we want the aspect ratios of our image to be exactly correct during training. We achieve this by cropping the training images to the exact aspect ratio as defined by the annotation  $y_I^R$ . We create minibatches for training by zero-padding all training images to a quadratic shape. The complete process is visualized in Fig. 6.

## Goals and Metrics

In this work, we aim to produce a series of images that show a specific car, i.e., a specific model with a specific body style and a certain color, in different rotations. In particular, we aim to create a series of images that exhibits continuous rotation of the car of interest. Evaluating a system that produces such a series of images, however, is non-trivial as we need to measure an abundance of aspects in the resulting images. First, we obviously want the generated images to be of high quality, i.e., the images should look realistic. Note that this requirement should ideally hold for all possible conditions, i.e., we also want to produce high quality for combinations of target labels that do not exist in the dataset. Here, we specifically aim to produce continuous rotations despite the fact that our dataset is only annotated with discrete rotation labels. In "Image Quality", we discuss metrics that capture these aspects in the generated images.

In addition to image quality, we expect the generated images to follow our conditioning guidance, i.e., the generative model should produce images that show a car that exactly fulfills our specifications. These specifications include all annotations in our dataset, i.e.,

1. body color
2. car manufacturer
3. car model
4. rotation
5. body style
6. background scenery
7. image aspect ratio

. In "Conditioning" we introduce metrics that measure the realization of these specifications.

Our goal is to rotate a specific car in a specific scenery. As such, it is important to maintain all conditions across a series of different rotations, i.e., ideally, the only aspect that changes in a series is the rotation/orientation of the car. Besides that, it should still be the same car in the same scene. In "Conditioning" we discuss measuring the quality and consistency of conditions under different rotations.

Finally, we need to measure the quality of the rotation in a series of images. Here, we aim to measure two aspects. First, we need to find a measurement to properly quantify the deviation of generated rotations from our specified rotation. Second, ideally, we would like to perform complete rotation cycles, i.e., rotate from  $0^\circ$  to  $360^\circ$  as smooth and steadily as possible.

## Image Quality

In this section, we will shortly introduce the Fréchet Inception Distance (FID) for evaluating image quality. Evaluating the quality of generated images is highly non-trivial as any metric basically needs to replicate human perception of images. As such, there is no perfect method to do so. However,

*Fréchet Inception Distance* [13] is the most widely used measurement for quantitatively assessing the quality of a set of generated images. Note that, as the name states, FID is not a metric but a measurement of distance. Fréchet Distance [14], in its origins, is a measure of similarity between two curves. However, it can also be used to measure the similarity between probability distributions. Note that we can

interpret a set of images as samples from a probability distribution. Thus, Fréchet Distance can be used to approximate the distance between two datasets, i.e., two sets of images that are sampled from different distributions. The central idea of FID is to measure this distance not in RGB-space but instead use a semantic representation from an intermediate layer of an Inceptionv3 [15] network trained on the ImageNet [12] dataset. Note that quantitatively measuring the quality of a set of images as perceived by humans is inherently hard and is the topic of ongoing research. However, at the time of writing, FID is the most common measurement for assessing generative models and has been shown to correlate well with human perception. However, we want to clearly state that FID is not a metric. Due to the calculation of Fréchet Distance in the feature space of a deep network, it also highly depends on the data that was used for training this network, i.e., natural images from the ImageNet dataset. Thus, it is reasonable to assume that FID works best when assessing data that is similar to this dataset. Our dataset, similar to the ImageNet dataset, mostly shows a single object of interest and images are often captured in a natural environment. However, we also have a lot of images with uniformly colored backgrounds as well as many urban scenes. Thus, this measurement cannot be perfect for assessing the quality of our generated images.

Fréchet Distance  $d(X, Y)$  between two one-dimensional distributions  $X$  and  $Y$  is defined as

$$d(X, Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2, \quad (2)$$

where  $X$  and  $Y$  are univariate normal distributions.  $\mu$  and  $\sigma$  are the mean and standard deviation of the respective distributions. Assuming that samples from  $X$  and  $Y$  are deep feature representations, we can calculate FID as the multivariate case of Eq. 2:

$$FID(X, Y) = \|\mu_X - \mu_Y\|^2 + Tr\left(\left(\sqrt{\Sigma_X} - \sqrt{\Sigma_Y}\right)^2\right), \quad (3)$$

where  $\Sigma_X$  and  $\Sigma_Y$  are the covariance matrices of  $X$  and  $Y$ .  $Tr$  is the trace, i.e., the sum of the diagonal values of a matrix. Note that  $\sqrt{\Sigma}$  refers to the matrix square root of  $\Sigma$

$$FID(X, Y) = \|\mu_X - \mu_Y\|^2 + Tr\left(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X\Sigma_Y}\right). \quad (4)$$

Although Eq. 3 is equivalent to Eq. 4, the second is the common formulation in literature because it requires less computation. Note that FID has various shortcomings. First, we have to assume that both  $X$  and  $Y$  are normal distributions. This is most likely not correct. However, by calculating feature statistics like mean and covariance, we effectively fit a normal distribution to our data. Thus, at the very least, we have to assume that this distribution is a proper approximation for the actual distributions  $X$  and  $Y$ . Second, Chong and

Forsyth [16] empirically found that FID linearly depends on the sample size. As a result, FID scores are not comparable across sample sizes as well as across different image resolutions. Our simple solution is to always calculate FID with images on a fixed resolution of  $256 \times 256$  pixels. We compare image sets that each consist of 50,000 images. In the following, we will simply refer to this measure as *FID50k*.

## Conditioning

In this section, we discuss measuring the realization of our specifications in the generated images, i.e., how accurate is the generative model in generating images that comply with our specifications.

**Conditioning Accuracy** In order to measure compliance with our specifications, we define the conditioning accuracy (CA). With CA, we implement a quantitative method that allows us to look and rate the generated images from a specification standpoint. We achieve this by learning a simple CNN with the target of classifying images into our various annotations. This can be achieved in a straight forward fashion by training on the annotated data from our dataset. We then use the resulting model to assess the generated images. If a generated image complies with our specification, then the classification model should simply output our specification. In case, the generated image does not comply with our specification, the output of our classification model will be different from our specification. CA is then simply the accuracy of the classification model. In order to rely on this metric, we have to assume that the classification model is sufficiently good in its task of classifying images according to our annotations. If this is the case, CA, which we compute as the accuracy of the generative model in combination with the classification model, now mainly depends on the contents of the generated images. In order to capture all possible specifications as defined in our dataset, we train individual<sup>1</sup> classifiers for each of the annotations, i.e., body color, car manufacturer, car model, rotation, body style, background scenery, and image aspect ratio.

**Conditioning Consistency** When considering a series of images on a predefined interpolation path, we often aim to only alter a single specification. Typically, this is the rotation of a car. In that case, we expect that this is the only aspect that changes in the images. To capture this, we again use the trained classifiers from the previous paragraph. However, instead of calculating the accuracy, we capture the consistency of all unchanged specifications. We do this by measuring the standard deviation  $\sigma$  of the output of the classifiers that we learned for CA across

<sup>1</sup> We also learned a single classifier for all annotations which, however, was inferior to multiple individual classifiers.

individual interpolation paths and then average standard deviations across multiple interpolation paths.

## Random Rotations and Latent Space Interpolation

In this section, we describe metrics suitable for the evaluation of mixed conditions. Mixed conditions, in our sense, are intermediate target labels that do not exist in the annotations of our dataset. However, due to the nature of our generative model, we can still input such intermediate specifications. This can be useful for multiple purposes. First, we can generate images with unseen specifications, e.g., generate a convertible version of a car model that is not available as a convertible. Second, we can interpolate between different specifications, e.g., interpolate from an off-road vehicle to a convertible. Similarly, we can interpolate between the discrete object rotation annotations in our dataset to create a complete series of images that shows a full rotation cycle of a car.

**FID for Random Rotations** When rotating objects, we expect the generated images to maintain high image quality. In "Image Quality", we already discussed FID as a metric for image quality. Applying FID to rotated objects in images is just as reasonable as calculating it on images that are generated with our quantized rotation conditions that we draw from the dataset. However, intermediate rotations between these quantizations do not exist in our dataset. As such, we expect image quality to be worse than on standard conditions. In order to capture this effect, we calculate FID with random rotations separately from the standard FID. Here, we draw conditions randomly from our dataset. Given such a set of conditions, we then choose a target rotation randomly. We then calculate FID between our dataset and a set of images generated with random target rotations.

**Learned Perceptual Image Patch Similarity (LPIPS)** LPIPS [17] is a measurement for image patch similarity. It is calculated as a weighted distance between deep features for individual image patches. The distances of the patches are then used to learn a small neural network that maps them to a single value. In *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric* the creators of learned perceptual image patch similarity (LPIPS) show that such a metric consistently outperforms traditional measures like the structural similarity index measure (SSIM) and L2-Distance in estimating the perceived distance of image patches. Note that patch similarity can be trivially applied to measure image similarity by simply increasing patch size such that it matches the image size. In this case, LPIPS between two images  $x_1$  and  $x_2$  can be calculated as

$$d(x_1, x_2) = \sum_{l \in \text{layers}} \frac{1}{H_l W_l} \sum_{h,w}^{H_l, W_l} \left\| w_l \odot (y_1^{l,h,w} - y_2^{l,h,w}) \right\|_2^2, \quad (5)$$

where  $y_1^l$  and  $y_2^l$  are the deep feature representations of  $x_1$  and  $x_2$  respectively after layer  $l$ .  $H_l$  and  $W_l$  represent the spatial dimensions of the representations after layer  $l$ .  $w_l$  is a layer-wise scaling factor for individual representations. Based on the distances, Zhang et al. then supervise a small network that is supposed to replicate human perception of image patch similarity. To learn this network, they collect a dataset that is composed of triplets of images. Each triplet consists of a reference image as well as two distorted versions of this image. They then ask humans to decide which of the distortions is more similar to the reference image. The distortions are sampled from a wide variety of traditional distortions that include photometric changes to image colors as well as noise and blur distortions. Other distortions include spatial changes like affine transformations as well as JPEG compression artifacts. A second set of distortions is sampled from various CNN-based methods in order to include typical artifacts generated by common CNNs. The metric learning task is then modeled as a binary classification problem where similar patches are supposed to result in a distance of zero and dissimilar patches are supposed to result in a distance of one. They show that such a learned metric correlates very well with human perception of image distance.

**Perceptual Path Smoothness (PPS)** When rotating objects, we aim for a rotation that is as smooth as possible, i.e., each fixed size interpolation step in the latent space should result in a constant rotation of the object in the image. To better capture this effect, we define the perceptual path smoothness (PPS)  $s_Z$ :

$$s_Z = \sigma \left[ \frac{1}{\epsilon^2} d(G(\text{slerp}(z_1, z_2; t)), G(\text{slerp}(z_1, z_2; t + \epsilon))) \right] \quad (6)$$

that measures variance along interpolation paths. Note that PPS is based on the learned LPIPS metric, i.e., like FID and various other measurements, both are just approximate indicators of generator performance and need to be considered in conjunction with other metrics as well as qualitative analysis. A downside of PPS is that rotation is not measured directly, but, instead it measures changes in the image. To at least partly overcome this issue, we only measure PPS on images that are generated with a constant black or white background. By this, we remove the dependency of PPS on background features.

**Measuring Continuous 3D Rotation** Note that rotation is just another label in our dataset. In this work, however, we specifically target good rotations. Thus, in addition to simply measuring rotations with CA. We explore an additional method that allows measuring of continuous rotation. In 3D

*Bounding Box Estimation Using Deep Learning and Geometry* [18], Mousavian et al. present a model that is capable of detecting 3D bounding boxes in two-dimensional images. Their model is learned on the KITTI [19] dataset which is a dataset of urban street scenes. The authors specifically target the detection of vehicles in those scenes. Using the resulting 3D bounding boxes, it is possible to approximate an object's rotation with respect to the camera. Note that, in theory, this requires a properly calibrated camera. However, we tested the algorithm on a small test set of our data and found that the object rotation estimates are very accurate. Note that we do not require high precision measurements for rotations and also do not expect to be able to generate objects that are precisely oriented in a specific angle. Instead, we only expect approximately correct orientations and continuous rotations. Given a series of images that show a full rotation cycle, we use the estimated rotations to evaluate the generated rotations. Here, we calculate an average distance between our specification and the rotation estimated by the 3D bounding box detector. Note that we found that the 3D bounding box detector often cannot distinguish between rotations that are  $180^\circ$  apart from each other. To mitigate the effects of these errors, we map all rotation estimates to a range between  $0^\circ$  and  $179^\circ$ . During qualitative analysis, we have never found an example in which our model confuses rotations. If we measure conditional rotation accuracy on the quantized rotations, all models consistently achieve an accuracy above 95%. Thus, we argue that such errors almost solely stem from the bounding box estimation, i.e., we get a better measurement of rotation quality by compensating for these errors.

## Conditional Object Synthesis

In this work, we aim to generate a series of images. Each of these images is supposed to show the same car in the same scene. However, we want to control the orientation of the car in the series. In particular, a series of images is supposed to show a full continuous rotation cycle of the car.

### Categorical Generator Conditioning

Conditioning the generator on the annotations from our dataset is relatively straight forward. We simply extend the latents  $z \in Z$  with the one-hot encoded conditions. However, our annotations are not independent, e.g., car manufacturer and car model are related. Thus, we cannot just draw random specifications during training. Instead, in order to get realistic specifications, we draw them from our dataset. Learning a conditional generator means that the discriminator also needs to provide conditional training feedback. We adopt a class-wise discrimination strategy, i.e., in addition to the standard

GAN loss, we employ additional GAN losses that provide learning feedback for individual classes. We effectively ask the discriminator to distinguish between, e.g., *real mercedes* and *fake mercedes* or *real countryside* and *fake countryside*. We do this for all annotated classes and gate the gradient during backpropagation such that only the relevant classes are considered. That is because the answer to *real or fake mercedes?* is useless if the currently considered image displays a car from another manufacturer. Another issue is that our data is only sparsely annotated, i.e., for most images, we do not have all attributes annotated. Thus, our training procedure must be able to handle these sparse annotations. For one-hot encoded target specifications this can be achieved by simply feeding an all-zero vector to  $G$  when a certain attribute is not annotated and ignoring the respective outputs of  $D$  in the loss calculation for  $G$  and  $D$ , e.g., if we do not know whether the image is supposed to show a *mercedes*, asking whether it shows a *real or fake mercedes* is unreasonable. Thus, we simply do not ask. Let  $m(a, b)$  be a derivable distance metric between  $a$  and  $b$ . We give a general error function  $\mathcal{L}^{D,\text{gan}}$  for a discriminator  $D$  as well as a general error function  $\mathcal{L}^{G,\text{gan}}$  for a generator  $G$  in Eqs. 7 and 8, respectively:

$$\mathcal{L}^{D,\text{gan}} = m(D(G(z)), 0) + m(D(\mathbf{x}), 1), \quad (7)$$

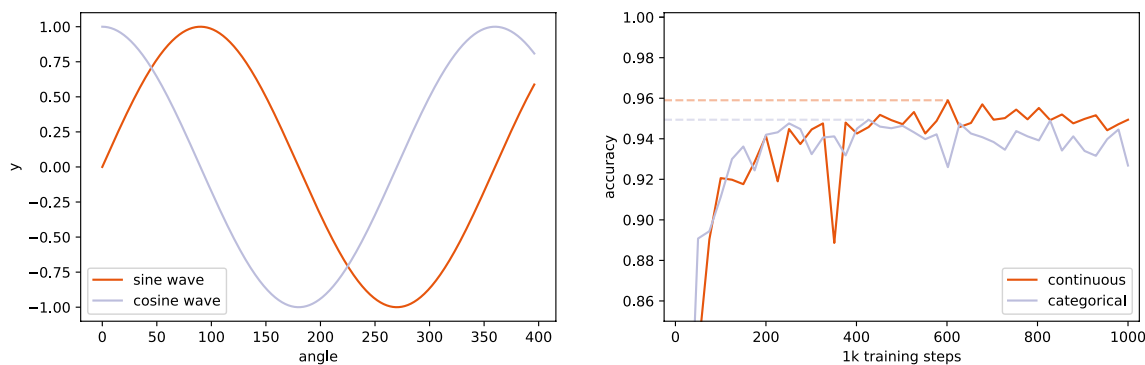
$$\mathcal{L}^{G,\text{gan}} = m(D(G(z)), 1). \quad (8)$$

Here, we use the standard sigmoid cross-entropy error for the distance metric  $m$ . We extend the general error functions from Eqs. 7 and 8 to error functions that are defined for all of our specifications. Let,  $y_a^t$  be a target specification for a single attribute  $a$ , i.e.,  $y_a^t$  is a one-hot encoded target attribute, e.g.,  $y_{color}^t$  is a one-hot vector with 11 entries. Each of these entries corresponds to one of the possible target colors. An entry  $y_{a,c}^t$  in  $y_a^t$  equals one if we want the synthesized image to show the corresponding characteristic  $c$ . Otherwise  $y_{a,c}^t$  is zero.  $y^t$  then is a multi-hot vector that consists of the one-hot vectors that represent all individual attributes, i.e.,  $y^t$  is a representation of our specification. Similarly,  $y$  is a multi-hot representation of the annotation of a training sample  $\mathbf{x}$ . Note that due to the sparse annotations in our dataset, individual components  $y_a$  of  $y$  are not always one-hot vectors but may also be all-zero vectors. We define the class-wise loss  $\mathcal{L}^{D,\text{gan}}$  for discriminator  $D$  in Eq. 9 and the class-wise loss  $\mathcal{L}^{G,\text{gan}}$  for generator  $G$  in Eq. 10:

$$\mathcal{L}_i^{D,\text{gan}} = m(D(G(z))_i, 0) \cdot y_i^t + m(D(\mathbf{x})_i, 1) \cdot y_i \quad (9)$$

$$\mathcal{L}_i^{G,\text{gan}} = m(D(G(z))_i, 1) \cdot y_i^t \quad (10)$$

Here,  $i$  iterates all individual characteristics for all possible attributes in  $y$  and  $y^t$ . Since  $y$  and  $y^t$  are zero for all classes that are currently irrelevant, the gradient for these classes



**Fig. 7** Left: the combination of *sine* and *cosine* yields a unique tuple representation for all possible angles. Right: learning orientation with a regression model (continuous) actually improves performance when

compared to a classification model trained on categorical annotations. Maximum performance for each model is highlighted by the dashed lines

is also zero. The full adversarial loss for  $D$  and  $G$  then is simply the mean of all individual components of  $\mathcal{L}^{D, \text{gan}}$  and  $\mathcal{L}^{G, \text{gan}}$  respectively. This procedure is reasonable for all attributes except for the orientation annotations as we will discuss in the next section.

### From Categorical Orientation to Continuous Rotation

The input of the convolutional stream of StyleGAN2 is a constant. Thus, as we already discussed in "StyleGAN2", the dense stream controls the output of the generator. Conditioning the generator of StyleGAN2 can be trivially achieved by stacking a multi-hot encoded representation of the categorical specifications to the input of the dense stream, i.e., by extending the style with the desired specifications. However, even though we only annotated categorical rotations in our dataset, we still aim for continuous rotation. Rotation, by definition, is inherently continuous as well as periodic, i.e., an angle  $\alpha \in [0^\circ; 360^\circ]$  is continuously defined and  $\alpha = \alpha + 360^\circ \forall \alpha \in \mathbb{R}$ . This behavior is perfectly implemented by both the *sine* and the *cosine* function, i.e.,  $\sin \alpha = \sin (\alpha + 360^\circ)$  and  $\cos \alpha = \cos (\alpha + 360^\circ)$ . However, both *sine* and *cosine* do not result in distinct values for individual angles  $\alpha$ , e.g.,  $\sin 45^\circ = \sin 135^\circ$ . Thus, we cannot simply use one or the other, but instead, need to encode the orientation by both *sine* and *cosine* of the angle  $\alpha$  to resolve these ambiguities. Such a formulation explicitly encodes continuity and periodicity of rotations and thus, in theory, should be superior to a categorical representation of rotations. In order to proof this theory, we conduct two experiments. First, we train a simple deep classifier on the categorical rotations. Second, we train a regression model with identical network architecture on the continuously encoded rotations. For both models, we measure classification accuracy. This can be trivially computed for the classification model. For the regression model, we simply quantize

each rotation estimate to the nearest categorical label. Both models use our discriminator architecture which additionally allows us to verify that the rotation target can be properly learned by it. This is obviously important because the discriminator creates the supervisory signal for learning the generator during training. Figure 7 visualizes validation performance of both models. We see that the regression model outperforms the classification model by approximately one accuracy point which equates to a 19% relative reduction in classification error. In addition, it seems to be less prone to overfitting as the validation performance towards the end of the training does not drop as fast as with the classification model.

In summary, using a continuous representation for object rotations seems to be advantageous as it explicitly encodes the natural ordering and periodicity of rotations. In addition, it is easier to learn for our discriminator. Thus, we opt for such a continuous representation throughout all experiments.

A continuous representation of rotations, however, cannot be learned in a standard adversarial setting, because it is unclear how to differentiate between real continuous rotations and fake continuous rotations in the discriminator. However, in an adversarial setting, we can see the generator as a model that strives to produce images that produce a specific output in the discriminator. Thus, we define a loss function for the generator that fulfills this requirement.

Given a target rotation label  $y_i^{f, \text{rot}}$  and latents  $z \in Z$ , we calculate the rotation error  $\mathcal{L}^{G, \text{rot}}$  for generator  $G$  as

$$\mathcal{L}^{G, \text{rot}} = \sum_i \left\| y_i^{f, \text{rot}} - D(G(y_i^{f, \text{rot}}, z)) \right\|_2^2, \tag{11}$$

where  $i \in \{0; 1\}$  iterates the two regression targets (*sine* and *cosine*) and the corresponding outputs of discriminator  $D$ , i.e., given a target rotation, we aim to produce an image for which the discriminator confirms this rotation.  $\mathcal{L}^{G, \text{rot}}$ , however, is only useful if the discriminator is actually able

to properly estimate object rotation. This can be learned on the real data. Given a real image  $x^r$  and corresponding rotation ground truth  $y^{r,rot}$ , we define the rotation loss  $\mathcal{L}^{D,rot}$  for discriminator  $D$  as

$$\mathcal{L}^{D,rot} = \sum_i \left\| \left| y_i^{r,rot} - D(x^r)_i \right| \right\|_2^2 \tag{12}$$

Again, orientation is not annotated for all images. Thus, in such cases, similarly to the categorical attributes, we set both *sine* and *cosine* inputs of  $G$  to zero and ignore the rotation in the loss calculation for  $G$  and  $D$ . Here it is important to choose a distinct combination of *sine* and *cosine* values that cannot conflict with any annotated orientations. Setting both to zero satisfies this requirement as none of the possible rotation angles  $\alpha$  can result in both  $\sin \alpha = 0$  and  $\cos \alpha = 0$ .

### Semi-supervised Cooperative Rotation Learning

Our dataset only labels quantized rotations, i.e., rotations have been quantized to the nearest  $45^\circ$ . While this makes the annotation process easy and even feasible at all, it results in two issues. First, multiple different rotations get quantized to the same orientation, i.e., we do not know the exact rotation. Second, there are no annotated images for intermediate rotations. The first issue can be solved by simply not requiring exact regression values. This is already implemented by the squared error in Eqs. 11 and 12 because the gradient of the squared error function diminishes when the rotation output is close to the desired target rotation, i.e., the impact on the overall error is lower the closer we are to the target rotation. Thus, estimated rotations only need to be in close vicinity to the quantized target rotation.

For the second issue, we exploit the natural order of our continuous rotation representation. Given that the input to our generator can be any rotation angle, we simply let it generate intermediate rotations. In order to properly estimate the rotation in a generated image that shows such an intermediate rotation angle, we need the discriminator to also learn those rotations, i.e., we can keep the loss formulations given in Eqs. 11 and 12 and add an additional rotation loss  $\mathcal{L}^{D,semi}$  that learns rotation from the generated fake images, i.e.,

$$\mathcal{L}^{D,semi} = \sum_i \left\| \left| y_i^{f,rot} - D(G(y_i^{f,rot}, z))_i \right| \right\|_2^2, \tag{13}$$

where we randomize  $y^{f,rot}$  in 20% of the cases by sampling a random angle between  $0^\circ$  and  $359^\circ$ . We do the same for  $\mathcal{L}^{G,rot}$ . As a result, for unlabeled intermediate angles, both networks learn to agree on a specific object rotation.

### Empirical Loss Scheduling

In preliminary experiments, a common issue was that the generator did not learn to create full rotation cycles, i.e., between some of the annotated  $45^\circ$  steps the car would collapse and then, from the collapsed image, an image that shows a car close to the next step would arise. We have also found that rotation is learned very early in the training. While the rotations become smoother and cleaner when training progresses, the generator never recovers from fundamental issues like collapsing cars and orientations that are fundamentally wrong. This observation is consistent with the findings of Karras et al. [1] who also observed that the fundamental composition of the generated images, i.e., the low frequency parts, are learned very early. Given these observations, we conclude that we need to enforce the learning of rotations early. However, learning our model is driven by multiple losses. Here, we will first discuss the discriminator loss:

$$\mathcal{L}^D = \lambda^{D,gan} \mathcal{L}^{D,gan} + \lambda^{D,rot} \mathcal{L}^{D,rot} + \lambda^{D,semi} \mathcal{L}^{D,semi} \tag{14}$$

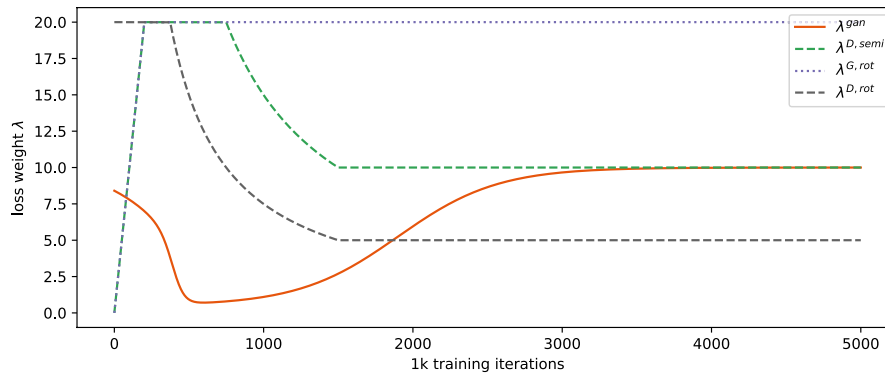
which is a combination of many losses that need to be balanced properly. Here,  $\lambda$  gives the weight of the individual losses. Note that it is the knowledge of the discriminator  $D$  that drives the learning of the generator  $G$ . Thus, before  $G$  is able to learn about rotations, we require it to have rudimentary image generation capabilities, i.e., in the very beginning, learning has to be driven by the GAN-Loss  $\mathcal{L}^{gan}$ . However, once  $G$  is capable of the aforementioned rudimentary image generation,  $G$  needs to learn rotation. Thus, we reduce  $\lambda^{gan}$  and simultaneously increase rotational learning for  $G$ . Note that  $D$  learns rotation from real images from the very beginning (see Eq. 19). Finally, when  $G$  has learned to create properly oriented objects, we can scale  $\lambda^{gan}$  back up and learn to improve image quality. This intuition results in the following schedule for  $\lambda^{gan}$ . At the current training progress  $i$  as measured by the number of thousands of images seen by the discriminator, i.e.,  $i = 1$  after having seen a thousand images, we calculate  $\lambda^{gan}(i)$  as a combination of two functions as given below. Here,  $\gamma(i)$  is a mixing function for the two components  $\iota(i)$  and  $\kappa(i)$ :

$$\gamma(i) = \sigma \left( \frac{10.0}{\beta} (i - \beta) \right), \tag{15}$$

$$\iota(i) = \sigma(\phi'(-i + \beta)), \tag{16}$$

$$\kappa(i) = \sigma(\phi^\kappa(i - \beta - \omega)), \tag{17}$$

where  $\sigma$  is the sigmoid function.  $\beta$  is an approximate offset at which  $\kappa(i)$  begins to dominate the expression.  $\phi'$  and  $\phi^\kappa$  control the steepness of the two functions and  $\omega$  controls the width of the valley at the intersection between  $\iota(i)$  and  $\kappa(i)$ .



**Fig. 8** Loss schedules for semi-supervised rotation estimation and rotation generation. In the beginning, it is important facilitate image generation, i.e., we need high  $\lambda^{\text{gan}}$ . Simultaneously,  $D$  needs to learn rotation, i.e., we need high  $\lambda^{D,\text{rot}}$  in the beginning. We need to enforce proper rotation in  $G$  throughout the training, i.e., we need a high  $\lambda^{G,\text{rot}}$  at all times. In the beginning, learning in a semi-supervised fashion hampers the training progress because  $G$  only produces random noise

Given the two individual functions  $\iota(i)$  and  $\kappa(i)$  as well as the mixing function  $\gamma(i)$  and desired minimal value  $\eta^{\text{gan}}$  and desired maximum value  $\zeta^{\text{gan}}$ , we calculate

$$\lambda^{\text{gan}}(i) = (\zeta^{\text{gan}} - \eta^{\text{gan}})[(1.0 - \gamma(i))\iota(i) + (\gamma(i)\kappa(i))] + \eta^{\text{gan}}. \tag{18}$$

We set  $\beta = 400$ ,  $\phi^i = 0.004$ ,  $\phi^k = 0.003$ ,  $\omega = 1500$ ,  $\eta^{\text{gan}} = 0.5$  and  $\zeta^{\text{gan}} = 10.0$  which results in the loss schedule shown in orange color in Fig. 8. Note that this schedule is identical for both the generator  $G$  as well as the discriminator  $D$ .

As mentioned above,  $D$  learns rotation from real images from the very beginning. However, as  $G$  begins to learn the quantized rotations that are annotated in our dataset, we need to reduce the importance of this loss in order to facilitate the learning of intermediate and continuous rotations by semi-supervision. The schedule

$$\lambda^{D,\text{rot}} = \min \left( \max \left( \frac{\beta^{D,\text{rot}} \eta^{D,\text{rot}}}{i + 1}, \eta^{D,\text{rot}} \right), \zeta^{D,\text{rot}} \right) \tag{19}$$

results in the dashed gray schedule shown in Fig. 8. Here, we set  $\beta^{D,\text{rot}} = 1500$ ,  $\eta^{D,\text{rot}} = 5.0$  and  $\zeta^{D,\text{rot}} = 20.0$ , i.e.,  $\lambda^{D,\text{rot}}$  starts at a high value of 20 and then decays to 5 after 1.5 million training images.

Similarly, we define

$$\lambda^{D,\text{semi}} = \min \left( \max \left( \frac{\beta^{D,\text{semi}} \eta^{D,\text{semi}}}{i + 1}, \eta^{D,\text{semi}} \right), \zeta^{D,\text{semi}}, i\phi^{D,\text{semi}} \right). \tag{20}$$

We set  $\beta^{D,\text{semi}} = 1500$ ,  $\eta^{D,\text{semi}} = 10.0$ ,  $\zeta^{D,\text{semi}} = 20.0$  and  $\phi^{D,\text{semi}} = 0.1$  which results in the green loss schedule shown in Fig. 8. In the beginning,  $G$  is not able to produce

images, i.e., trying to estimate rotation from such images is pointless. However, fundamental image composition that includes rotation is learned very early, thus we need to increase  $\lambda^{D,\text{semi}}$  fast and decrease  $\lambda^{\text{gan}}$  until rotations are learned properly. After rotation is learned, we can reduce rotation learning in  $D$  and increase importance of the GAN training to improve image quality

anything meaningful. Learning rotation from the early images produced by  $G$  hampers the training. Thus, with increasing image quality, we scale up the impact of the semi-supervision for  $D$ . After some time, both  $G$  and  $D$  have agreed on the intermediate rotations which allows us to reduce  $\lambda^{D,\text{semi}}$  towards the later parts of the training and thus, allow for a higher importance of the GAN training.

Similarly to  $\mathcal{L}^D$ , the generator loss  $\mathcal{L}^G$  is a combination of a GAN-Loss  $\mathcal{L}^{G,\text{gan}}$  and the rotation-based loss  $\mathcal{L}^{G,\text{rot}}$ .

$$\mathcal{L}^G = \lambda^{G,\text{gan}} \mathcal{L}^{G,\text{gan}} + \lambda^{G,\text{rot}} \mathcal{L}^{G,\text{rot}}, \tag{21}$$

$\lambda^{G,\text{gan}}$  follows the schedule given in Eq. 18. In the beginning,  $G$  cannot learn proper rotations from  $D$  because  $D$  needs to learn that first. Thus, with increased rotational knowledge of  $D$ , we can scale up rotational learning in  $G$ . However,  $G$  can learn to create continuous rotations because its learning of rotation does not depend on the quantized annotations of our dataset. Instead,  $G$  learns from  $D$ . Thus, we do not need to reduce  $\lambda^{G,\text{rot}}$  towards the end of the training. We calculate  $\lambda^{G,\text{rot}}$  as

$$\lambda^{G,\text{rot}} = \min(i * \phi^{G,\text{rot}}, \zeta^{G,\text{rot}}). \tag{22}$$

Here, we set  $\zeta^{G,\text{rot}} = 20.0$ ,  $\phi^{G,\text{rot}} = 0.2$  which results in the dotted loss schedule shown in Fig. 8.

## Results

In this section, we detail and discuss our results both quantitatively as well as qualitatively.

**Table 1** Quantitative results for image quality

Model	#train img (m)	Train res	↓FID50k	↓FID50k random rotation	↑quantized rotation acc (%)	↓avg. rotation distance	↓PPS
Coop.	5	256	7.27	7.46	<b>99.8</b>	19.23°	<b>0.0069</b>
Coop. + semi	5	256	5.70	6.76	98.5	<b>18.71°</b>	0.0113
Coop. + semi + class <i>D</i>	5	256	<b>5.43</b>	<b>6.27</b>	96.7	21.48°	0.0099
Coop. + semi + class <i>D</i>	5 m	512	5.15	5.40	<b>96.3</b>	<b>22.70°</b>	<b>0.0087</b>
Coop. + semi + class <i>D</i>	18 m	512	<b>3.53</b>	<b>3.64</b>	93.33	23.93°	0.0101

Starting with a baseline model that uses StyleGAN2’s standard loss definition and our rotation target (top) we improve by adding semi-supervision to the rotation training. Changing StyleGAN2’s standard loss to our class-wise discrimination strategy improves the model even further. Bottom: our model trained on a higher resolution

Bold means best for given resolution (“Train res”)

**Table 2** Quantitative results for conditioning accuracy

Model	#train img (m)	Train res	↑model (%)	↑manufacturer (%)	↑color (%)	↑body (%)	↑background (%)	↑ratio (%)
Coop.	5	256	48.05	70.09	<b>69.96</b>	<b>79.06</b>	<b>90.92</b>	<b>99.31</b>
Coop. + semi	5	256	<b>52.26</b>	73.74	68.83	78.32	89.92	98.61
Coop. + semi + class <i>D</i>	5	256	47.81	<b>77.25</b>	64.60	78.36	83.46	97.81
Coop. + semi + class <i>D</i>	5	512	46.99	78.15	57.62	78.65	83.84	94.97
Coop. + semi + class <i>D</i>	18	512	<b>71.62</b>	<b>90.55</b>	<b>66.33</b>	<b>78.94</b>	<b>86.71</b>	<b>97.61</b>

Higher values correspond to a higher level of control, i.e., higher accuracies mean that a model better adheres to our specifications

Bold means best for given resolution (“Train res”)

## Quantitative Results

In Table 1, we give quantitative results for image quality. Here, it is important to compare models at identical resolutions. For this work, we chose to compare models at an image size of  $256 \times 256$  pixels. In addition, we present results of our full model at a higher resolution of  $512 \times 512$  pixels that improves image quality. We see that both our semi-supervised rotation objective as well as our class-wise discrimination strategy improve image quality as measured by FID. Both also improve image quality for random rotations. However, our rotational measures generally seem to decrease with improved image quality. The negative effects however are marginal and we were not able to verify them qualitatively. Moreover, without our semi-supervised objective, objects frequently collapsed during a rotation cycle. Unfortunately, this is an effect that does not seem to be captured by the rotational measurements. It may be the reason for the considerable improvements in the *FID50k random rotation* measure. The improvements in overall image quality are immediately obvious in qualitative comparisons.

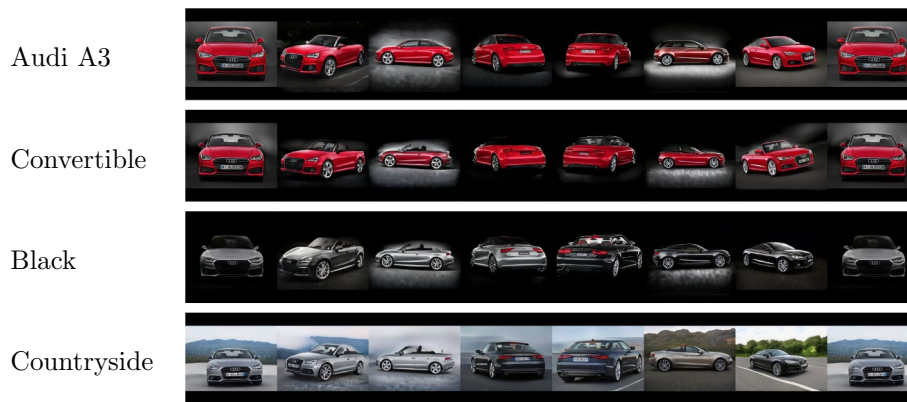
In Table 2, we show that our method generally adheres to our specifications. Note that accuracy values generally should be considered with the number of possible classes

in mind, i.e., a higher number of possibilities for a specific attribute naturally reduces accuracy. In other words, both the generative model and the classifier that we trained to assess the generated images generally have to solve a harder task when compared to attributes with a low number of classes. In addition, note that due to the nature of our measurements that use imperfect classifiers to assess the attributes of the generated objects small changes by a few percent cannot be indicative of relevant improvements. As such, we consider the *conditioning accuracy* for the attributes *car model* (67 classes), *body style* (10 classes), and *aspect ratio* (5 classes) to be almost identical across all methods that we analyzed. However, we see a significant improvement of around 7.1 absolute percentage points for the *manufacturer* (18 classes). This is equivalent to a 23% reduction in the conditioning error. On the other hand, simultaneously, we observe that color accuracy is reduced by 5.4% and background accuracy is reduced by 7.5%. The consistency measurements in Table 3 paint a similar picture. Here background consistency and aspect ratio consistency during rotation decreased with our full method. We also see a small reduction in color consistency. However, all other consistency measures improve considerably with our full model which matches our

**Table 3** Quantitative results for conditioning consistency, i.e., classifier standard deviation for rotating objects

Model	#train img	train res	↓model	↓manufacturer	↓color	↓body	↓background	↓ratio
Coop.	5m	256	0.01915	0.03834	0.0391	0.0617	<b>0.03609</b>	<b>0.00411</b>
Coop. + semi	5m	256	0.01911	0.04552	<b>0.03567</b>	0.06981	0.04480	0.01727
Coop. + semi + class <i>D</i>	5m	256	<b>0.01590</b>	<b>0.03240</b>	0.04031	<b>0.04945</b>	0.04458	0.01163
Coop. + semi + class <i>D</i>	5m	512	0.01460	0.03089	0.04029	0.04739	0.04350	0.01961
Coop. + semi + class <i>D</i>	18m	512	<b>0.01097</b>	<b>0.02159</b>	<b>0.03645</b>	<b>0.04234</b>	<b>0.04175</b>	<b>0.01454</b>

Lower values mean that objects are rendered more consistently when undergoing rotations  
 Bold means best for given resolution (“Train res”)



**Fig. 9** Qualitative example for rotations under changing specifications. Each line shows a full rotation cycle. Top: initial image showing an Audi A3 hatchback in red on a black background. Upper middle: initial image with *body style* changed from hatchback to convertible.

Lower middle: convertible with *body color* changed from red to black. Bottom: black convertible with *background scenery* changed from black to countryside

**Fig. 10** Qualitative examples for mixed conditions that are not in the dataset, i.e., out-of-sample results



qualitative observations the we will further discuss in the next section.

## Qualitative Results

In Fig. 9, we show examples of generated rotation cycles. We showcase control by changing individual attributes. The first row shows an initial image series. The second row shows the same series with the *body style* of the car changed to *convertible*. In the next row, we change the color of the convertible from *red* to *black*. Here, we can also observe a very common case of failure. The *body color* does not remain constant during the rotation cycle. This is a common problem when generating *silver*, *gray* and *black* cars. These colors seem to be particularly hard to differentiate and often get confused.

In the third row, we place the *black convertible* from the previous row in the *country side*. Here, we observe similar issues with the *body color*. However, the *background scenery* is properly changed. In Fig. 10, we show out-of-sample results based on combinations of attributes that are not available in the training data.

## Conclusion

In this work, we have proposed a method for control over image synthesis systems. Our method is able to handle data with multiple labels that is only sparsely annotated. We have also shown that, in an adversarial setting, a cooperative regression target that is jointly optimized by both generator and discriminator can be used for fine-grained and continuous control of object orientation. We have shown that such a cooperative target can be learned in a semi-supervised fashion which allows us to learn continuous orientations from quantized orientation annotations.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will

need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and improving the image quality of stylegan. In: 2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020. Computer Vision Foundation/IEEE. 2020. pp 8107–16. <https://doi.org/10.1109/CVPR42600.2020.00813>.
- Härkönen E, Hertzmann A, Lehtinen J, Paris S. Ganspace: discovering interpretable GAN controls. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H, editors. Advances in neural information processing systems 33: annual conference on neural information processing systems 2020, NeurIPS 2020, December 6–12, 2020, Virtual. 2020.
- Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville AC, Bengio Y. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Advances in neural information processing systems 27: annual conference on neural information processing systems 2014, December 8–13 2014, Montreal, Quebec, Canada. 2014. pp 2672–80.
- Xu H, Li C, Rahaman MM, Yao Y, Li Z, Zhang J, Kulwa F, Zhao X, Qi S, Teng Y. An enhanced framework of generative adversarial networks (ef-gans) for environmental microorganism image augmentation with limited rotation-invariant training data. IEEE Access. 2020;8:187455–69. <https://doi.org/10.1109/ACCESS.2020.3031059>.
- Li X, Zhengshun D, Huang Y, Tan Z. A deep translation (gan) based change detection network for optical and sar remote sensing images. ISPRS J Photogramm Remote Sens. 2021;179:14–34. <https://doi.org/10.1016/j.isprsjprs.2021.07.007>.
- Chan ER, Monteiro M, Kellnhöfer P, Wu J, Wetzstein G. Pi-gan: periodic implicit generative adversarial networks for 3d-aware image synthesis. In: IEEE conference on computer vision and pattern recognition, CVPR 2021, virtual, June 19–25, 2021. Computer Vision Foundation/IEEE. 2021. pp 5799–809.
- Zhang Y, Chen W, Ling H, Gao J, Zhang Y, Torralba A, Fidler S. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. In: 9th international conference on learning representations, ICLR 2021, virtual event, Austria, May 3–7, 2021. OpenReview.net. 2021.
- Tewari A, Elgharib M, Bharaj G, Bernard F, Seidel H, Pérez P, Zollhöfer M, Theobalt C. Stylerig: rigging stylegan for 3d control over portrait images. In: 2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020. Computer Vision Foundation/IEEE. 2020. pp 6141–50. <https://doi.org/10.1109/CVPR42600.2020.00618>.
- Shoshan A, Bhonker N, Kviatkovsky I, Medioni GG. Gan-control: explicitly controllable gans. In: 2021 IEEE/CVF international conference on computer vision, ICCV 2021, Montreal, QC, Canada, October 10–17, 2021. IEEE. 2021. pp 14063–73. <https://doi.org/10.1109/ICCV48922.2021.01382>.
- Brehm S, Harzig P, Einfalt M, Lienhart R. Learning segmentation from object color. In: 3rd IEEE conference on multimedia information processing and retrieval, MIPR 2020, Shenzhen, China, August 6–8, 2020. IEEE. 2020. pp 139–44. <https://doi.org/10.1109/MIPR49039.2020.00036>.
- He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June

- 27–30, 2016. IEEE Computer Society. 2016. pp. 770–78. <https://doi.org/10.1109/CVPR.2016.90>.
12. Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L. Imagenet: a large-scale hierarchical image database. In: 2009 IEEE computer society conference on computer vision and pattern recognition (CVPR 2009), 20–25 June 2009, Miami, Florida, USA. IEEE Computer Society. 2009. pp. 248–55. <https://doi.org/10.1109/CVPR.2009.5206848>.
  13. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R, editors. Advances in neural information processing systems 30: annual conference on neural information processing systems 2017, December 4–9, 2017, Long Beach, CA, USA. 2017. pp. 6626–37.
  14. Fréchet M. Sur la distance de deux lois de probabilité. *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences*. 1957;244(6):689–92.
  15. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: 2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016. IEEE Computer Society. 2016. pp. 2818–26. <https://doi.org/10.1109/CVPR.2016.308>.
  16. Chong MJ, Forsyth DA. Effectively unbiased FID and inception score and where to find them. In: 2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020. Computer Vision Foundation/IEEE. 2020. pp. 6069–78. <https://doi.org/10.1109/CVPR42600.2020.00611>.
  17. Zhang R, Isola P, Efros AA, Shechtman E, Wang O. The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018. Computer Vision Foundation/IEEE Computer Society. 2018. pp. 586–95. <https://doi.org/10.1109/CVPR.2018.00068>.
  18. Mousavian A, Anguelov D, Flynn J, Kosecka J. 3d bounding box estimation using deep learning and geometry. In: 2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017. IEEE Computer Society. 2017. pp. 5632–40. <https://doi.org/10.1109/CVPR.2017.597>.
  19. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition, Providence, RI, USA, June 16–21, 2012. IEEE Computer Society (2012). pp. 3354–61. <https://doi.org/10.1109/CVPR.2012.6248074>.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.