Pose-Conditioned Image Generation

Shubham Shrivastava
Department of Computer Science
Stanford University
shubhams@stanford.edu

Amir Ziai
Department of Computer Science
Stanford University
amirziai@stanford.edu

Abstract

Autonomous driving application requires detection and localization of objects within a few centimeters. Deep learning models that can achieve this level of accuracy require millions of training data samples. Manually annotating data at this scale is highly cumbersome and expensive. Furthermore, annotating object poses in 3D is not possible from just a single image without requiring either additional sensors or multi-frame non-linear optimization. In this work we present *PoseGen*, a methodology for generating a realistic image of an object given a desired pose, appearance, and background. We do this in an unsupervised way by conditioning the generation process on various attributes. We also release a dataset of paired objects and silhouette masks (*TeslaPoseGen* dataset) which we hope will help the research community further tackle this problem[1][2].

1 Introduction

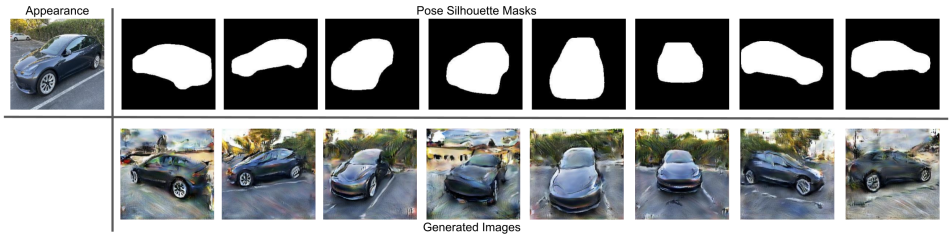


Figure 1: Pose-Conditioned Image Generation

Object Pose Estimation is a critical task for autonomous driving [1]. An emerging field of infrastructure assisted autonomous navigation [2] requires vehicles to be detected and localized within a few centimeters. Building such models requires tremendous amounts of training data. However, data collection can be extremely expensive and an unsupervised method of obtaining labeled data is desirable. With this work, we are exploring the possibility of generating realistic training data using generative models. The generated image and the input vehicle pose subsequently serve as training sample for the downstream task. The challenge is that we need to generate realistic and high resolution images.

To do this, we utilize concepts from conditional GANs and effectively encode various attributes such as background, object appearance, and object pose, into a latent vector, and then decode it into the target image. The desired pose is encoded as a binary silhouette mask, which can come from either a simulation engine [3], or a 3D model renderer [4] given the 3D model of the vehicle at the test time. During training, we extract the binary mask using a pre-trained Mask R-CNN [5], and that gives us paired RGB and binary mask data for the object of interest. Two additional encoders are used

[1] Code is available at: `https://github.com/towardsautonomy/PoseGen`
[2] *TeslaPoseGen* dataset is available at: `https://www.towardsautonomy.com/teslaposegen-dataset`

to encode object appearance and background respectively. Each encoder produces a latent vector, and are later concatenated into a composite latent vector on which the generator operates. Figure 1 depicts the process of generating images given desired poses for a single input (i.e. appearance) image.

2 Related work

Siarohin et al. [6] propose a method for generating images of humans given input pose and appearance. They do this by using deformable skip-connections in the generator of a GAN, which can transfer information between the encoder and the decoder as a function of the difference between the input and the desired poses. They also propose a novel nearest-neighbour loss function. This metric is computed as the minimum L_1 norm of vectorial representations (extracted using convolutional filters) of fixed-size patches in the input and generated images. They show that this loss is less sensitive than common pixel-level losses and it proves to improve the generation quality for this task. We also use skip connections in our architecture but do not use the deformable variety. Our loss function is a combination of GAN min-max loss and pixel-level reconstruction loss.

Men et al. [7] generate images conditioned on desired poses of humans using key landmarks. Jakab et al. [8] also generate images conditioned on landmarks. However, in their case they learn these landmarks in an unsupervised way. Our work is different in that landmarks are harder to obtain for 2D images of arbitrary objects such as cars. Instead, we use instance segmentation as a proxy to pose. Although not as accurate, this allows us to rapidly generate training data since we can run instance segmentation on images in order to collect the silhouette.

3 Technical Approach

Our proposal is largely inspired by [7]. They demonstrate the ability to generate realistic photos of subjects given an input pose and clothing style. In a similar way, we are interested in generating realistic images given a reference object, a pose, and a background image.

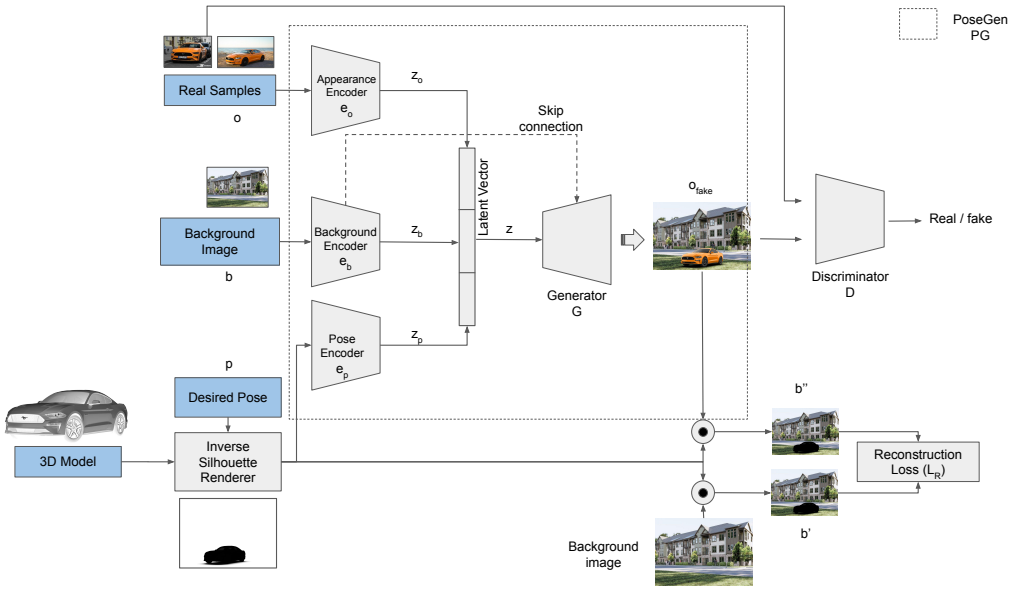


Figure 2: PoseGen architecture

3.1 General Case

Our proposed architecture is depicted in figure 2 and consists of three encoders: (1) object appearance encoder e_o, which simply encodes the appearance of the object o into a latent variable z_o ($z_o = e_o(o)$), (2) background encoder e_b, which encodes the background image b so that it can be used while

generating the image $z_b = e_b(b)$, and (3) pose encoder e_p, which encodes the binary mask representing the silhouette of the object at the desired pose, i.e. $z_p = e_p(p)$. These encoders entail a stack of CNNs, final outputs of which are flattened and concatenated to form a latent vector z. A generator G will then be used to generate an image, i.e. $o_{fake} = G(z)$. We dub this generation process PoseGen (PG):

$$o_{fake} = PG(o, b, p) \tag{1}$$

We use a discriminator D to distinguish real and fake images. [9] show that using hinge loss can be beneficial (improved IS [10] and FID [11]) for training GANs, which is what we use for this work:

$$\mathcal{L}_{GAN} = \mathcal{L}_G + \mathcal{L}_D \tag{2}$$
$$\mathcal{L}_G = -\mathbb{E}_z D(G(z)) \tag{3}$$
$$\mathcal{L}_D = \mathbb{E}_x \max(0, 1 - D(o)) + \mathbb{E}_z \max(0, 1 + D(G(z)) \tag{4}$$

where, o is a real image sampled from the training data distribution.

We can now define the total loss $\mathcal{L}$:

$$\mathcal{L} = \lambda_{GAN}\mathcal{L}_{GAN} + \lambda_R\mathcal{L}_R \tag{5}$$

$\mathcal{L}_R$ is the reconstruction loss between masked background image and the masked generated image:

$$\mathcal{L}_R = ||b \odot (1 - p) - o_{fake} \odot (1 - p)||_2^2 \tag{6}$$

where $\odot$ is the element-wise dot product, $o_{fake} = G(z)$, $1 - p$ is the inverse silhouette, $\lambda_{GAN} > 0$, and $\lambda_R > 0$ control the importance we place on each component of the loss.

The motivation for including this loss term is to encourage the generation process to reproduce the background with high fidelity. This requirement puts a lot of pressure on e_b for perfect reconstruction. Therefore, we added a skip connections from various e_b layers directly into the generator G, so that the background can be reconstructed easier if required.

Algorithm 1 Pseudocode for our proposed system representing training for a batch

Require: Sample a batch of o, p, b from training sets O_t, P_t, and B_t
1: Sample object model appearance: $o \sim sample(O_t)$
2: Sample object pose binary mask: $p \sim sample(P_t)$
3: Sample background image: $b \sim sample(B_t)$
4: $z_a = enc_o(o)$
5: $z_p = enc_p(p)$
6: $z_b = enc_b(b)$
7: $z = concat(z_a, z_p, z_b)$
8: $o_{fake} = G(z)$
9: $loss = \mathcal{L}(o, p, b, o_{fake})$
10: Back-propagate and train the model

Our training loop is described in Algorithm 1. Once training is completed we can generate new samples by providing the desired values o, b, and p.

3.2 Pose and Appearance Conditioned Image Generation

Another variant of our method is where, given paired samples of RGB image and the corresponding object mask, we would want to reconstruct the object very accurately while making sure that the image still looks realistic. Samples generated from this method will look realistic and will have similar background as observed in the train set, however the background will not be controllable. This variant only consists of two encoders instead of three. The objective of this variant is to generate an image of an object corresponding with the provided mask and a realistic background. Our reconstruction loss with this variant then becomes:

$$\mathcal{L}_R = ||o \odot p - G(z) \odot p||_2^2 \tag{7}$$

3.3 Pose-Conditioned Image Generation

One could also think about further simplifying this architecture and only condition the generative process on the pose, this would entail using binary mask as an input and generating the target RGB image as the output. This is possible in our framework since we have already extracted the binary masks corresponding to the object of interest using mask R-CNN [5]. However, a major limitation of this method would be that we can no longer control the type of object or the background. The simplified model seems to output higher quality images compared to the ones with additional conditional inputs and is discussed in the results section.

3.4 Binary mask extraction

We use the PyTorch [12] implementation of mask R-CNN [5] with a ResNet-50-FPN [13] backbone, which is pre-trained on COCO train2017 [14]. We filter out the segmentations that are not of the object of interest, apply a 0.5 threshold to binarize, and take the largest of all the surviving masks. We ended up removing a very small number of images from our datasets that yielded no mask when we applied this method.

4 Datasets

We used two datasets to study the performance of our proposed system.

4.1 Stanford Cars

The first dataset we explored is Stanford Cars [15], which has more than 16k images of 196 categories of cars. Categories are a combination of make, model, and year. This dataset was originally split into two roughly equal partitions. However, this stratification results in cars from the same category to appear in both partitions, which means that any model trained on the train set would have seen very similar examples of the test set. Therefore, we decided to stratify the dataset at the level of categories.

In order to accomplish this, we first combined the original train and test sets. Then we randomly partitioned the car models so that roughly 80% would land in training, 10% in validation, and 10% in test. The number of images for each category can range between 48 and 148. Therefore, this stratification translates into a total of 12,678 train images (78%), 2,075 validation images (13%), and 1,435 for test (9%).

4.2 TeslaPoseGen dataset

As noted in the previous section the Stanford Cars dataset has a limited number of images of the same car in different poses and with different backgrounds. Moreover, we wanted to be able to decompose the foreground car and the background image. Therefore, we collected our own dataset of the same Tesla car shot in two different locations with a variety of different poses. We collected 1,509 images with the car in the foreground and 736 images of the background (without the car). Furthermore, we extract binary masks corresponding to the object of interest as described in section 3.4.

This dataset provides us with an order of magnitude larger number of images from the same car relative to Stanford Cars, and can help us better understand the feasibility of the proposed methodology.

5 Results

In this section we present experimental results and analyses. Before diving in, we'll first explain the metrics we used for quantifying the proposed system. We also explain the baseline and oracles that can help us with contextualizing the results.

5.1 Metrics

FID [11], KID [16], and IS [10], are commonly used for measuring the quality of image generation models. In addition to those, we introduce two new metrics that are more specific to our task.

5.1.1 Intersection over Union (IoU)

As mentioned earlier, we use mask R-CNN [5] to extract the binary mask of the desired object's silhouette. Since we are interested in conditional generation given pose, we expect the binary mask of the generated image to accurately track that of the desired pose. An intuitive way to capture this similarity is to compute the IoU between the two binary masks:

$$IoU(bm_o, bm_f) = \frac{bm_o \cap bm_f}{bm_o \cup bm_f} \tag{8}$$

where bm_o is the binary mask for the input object and bm_f is the binary mask for the generated image.

Note that this value ranges between 0 and 1. Higher IoU values mean that the generated car's pose is closer to the desired pose. We get IoU=0 if instance segmentation fails to detect the object. The IoU values we report are the mean of individual IoU values across the dataset.

5.1.2 Inception cosine similarity (I_{sim})

Additionally, because we are also interested in conditioning on an input object, we need a way of measuring whether there's good visual correspondence between the input object and generated image. FID and KID can capture the overall similarity between the real and generated image sets as a whole, but fail to capture pairwise similarity. Therefore, we compute the cosine similarity between the inception embeddings of the real and generated image as follows:

$$I_{sim}(f_o, f_f) = \frac{f_o^T f_f}{||f_o||\,||f_f||} \tag{9}$$

where f_o and f_f are 2,048 dimensional vectors corresponding to the last pooling layer of the Inception V3 [17] model for the real object and the generated (i.e. fake) images respectively.

Note that this metric ranges between -1 and 1. Higher values mean that the the generated image looks more similar to the input. In cases where we're also conditioning on the background, we don't necessarily want this metric to be maximized. We considered extracting the embeddings of masked images (masked by pose) so we can capture visual similarity for background and object independently. However, it isn't clear whether this approach leads to reliable embeddings since Inception V3 was not trained with masked inputs. Thus, we ended up using unmasked images for our experiments.

Similar to IoU, we report the mean of I_{sim} values across the dataset.

5.2 Baselines

If successful, our methodology enables the generation of realistic images that honor the provided inputs. To the best of our knowledge, no prior work has attempted pose-conditioned generation using Stanford Cars; [18] only report metrics for unconditional generation.

5.2.1 Baseline: Random RGB

Probably the lowest bar of performance is to ensure that any generation method can handily beat randomly generated images. We simply generate random RGB images and compute all metrics on this baseline.

5.3 Oracles

It is very important to establish the upper bounds of what we can hope to achieve. This knowledge can help us prioritize our efforts. We capture these upper bounds through three related oracles.

5.3.1 Oracle 1: autoencoder

This method captures the capacity of the proposed methodology. In other words, given an input image, how well can we reconstruct it by first compressing it into a latent vector and then generating it from that vector? We used full image reconstruction loss for $\mathcal{L}_R$ instead of a masked one.

5.3.2 Oracle 2: random image from the train set

This method captures the upper bound on how realistic the generated images can be. For each generated image we simply compare it to an image which is randomly selected from the train set. By design, this method should serve as an upper bound on metrics that capture image sharpness and closeness of the generated set to the real set of images. So we can expect this method to provide an upper bound for FID, KID, IS, and I_{sim}.

5.3.3 Oracle 3: closest image from the train set by IoU

Since we are interested in conditioning on pose, we can extend oracle 2 to select the image from the training set that has the highest IoU with the real input image. With this method, we can expect an upper bound on all metrics.

5.4 Experimental procedure

We train all models for 75k epochs, evaluate every 500 epochs on the evaluation set, and report metrics on test set for the best (i.e. average of IoU and I_{sim}) evaluation results.

We generate 256×256 images, use $lr = 2 \times 10^{-4}$, batch size of 64, and $n_z = 256$.

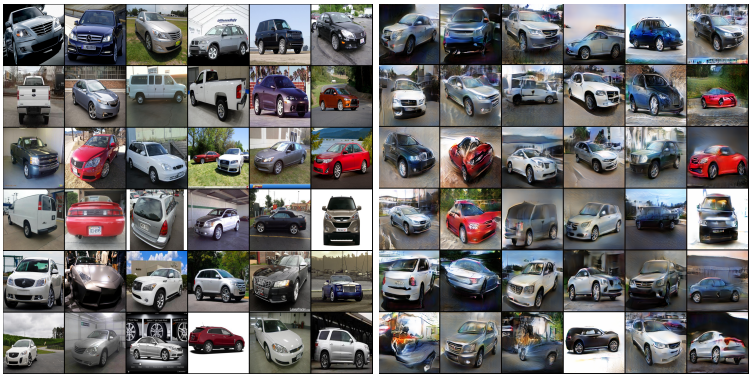
5.5 Experiments

5.5.1 Stanford cars

We started our experiments with the Stanford cars dataset. Although unconditional generation yielded reasonable results, we were not able to produce good conditional results. We partially attribute this to the dataset having a small number of images in each pose and decided to collect our own dataset (i.e. TeslaPoseGen) as previously discussed. The experiments we report in Table 1 were conducted before we conceived of I_{sim} and IoU so these metrics are not reported for all rows.

Table 1: Experiment results for the Stanford cars dataset.

Method	**IS**	**FID**	**KID**	**IoU**	I_{sim}
Baseline 1: random RGB	1.08	434.66	0.61	0.00	0.34
Oracle 1: autoencoder	2.75	57.63	0.04	-	-
Oracle 2: random image from train set	3.18	22.50	0.00	0.43	0.67
Oracle 3: closest train image by IoU	3.23	20.36	0.00	0.85	0.72
Unconditional generation	2.18	50.09	0.03	-	-



(a) Real (b) Fake

Figure 3: Real and Fake samples generated Oracle 1

This experiment gave us confidence that we can at least generate high resolution images as shown in figure 3. However, we pivoted to the Tesla dataset since we were not able to generate good conditional outputs for this dataset.

5.5.2 TeslaPoseGen

We present results for three settings in Table 2. The pose setting is only conditioned on the input pose. In other words, we use e_p to encode the pose into a latent vector z_p and $z = z_p$. For pose + car we use two encoders: e_p and e_o, and $z = concat(z_o, z_p)$. Finally, pose + car + background uses all three encoders, i.e. $z = concat(z_o, z_p, z_b)$.

Table 2: Experiment results for the TeslaPoseGen dataset.

Method	**IS**	**FID**	**KID**	**IoU**	I_{sim}
Baseline: random RGB	1.07	449.01	0.68	0.00	0.35
Oracle 1: autoencoder	1.00	259.96	0.38	0.40	0.68
Oracle 2: random image from the train set	1.58	37.96	0.00	0.43	0.78
Oracle 3: closest train image by IoU	1.48	27.70	0.00	0.87	0.91
Pose	**2.66**	**228.17**	**0.20**	**0.26**	**0.67**
Pose + car	1.47	363.99	0.51	0.00	0.54
Pose + car + background	1.00	366.53	0.61	0.03	0.60

All three setting are beating the baseline. Notice that the pose-only setup performs the best. The trend we notice is that using more input types results in lower generation quality. This is not very surprising- it is more challenging to combine more than one type of input into a coherent and realistic image. We see that as image sharpness declines, so does IS, and this results in a failure of instance segmentation, which explains the low IoU.

5.6 Qualitative Results and Analysis

In this section we will conduct a qualitative analysis of the TeslaPoseGen dataset. All analyses and samples shown in this section are selected from the test set once we finalized our experiments.

(a) Input silhouette maps (b) Generated Images

Figure 4: Input object binary silhouette masks and corresponding generated image from our method.

The examples in figure 4 are generated using the pose-only variant and are not cherry-picked. We noticed that many examples we looked at from this model seem to follow the pose very accurately, but the overall mean IoU number is only 0.26, which is much lower than the oracle.

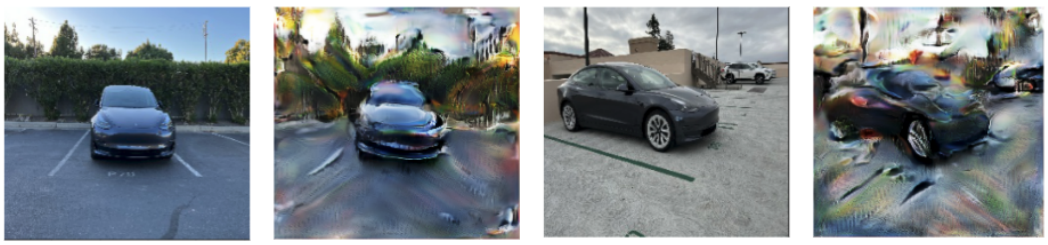


Figure 5: Sharp images are required for successful image segmentation. (b) is the generated image corresponding to provided image (a) and has IoU=0.88. (d) is the generated image corresponding to input image (c), where image segmentation fails and results in IoU=0.

We investigated examples where IoU is zero and discovered an interesting pattern. Figure 5 shows two pairs of input and generated images in the pose-only setting. We see that as generated image

sharpness degrades, we reach a point where the image is no longer recognizable by mask R-CNN and we get IoU=0. Looking at these images it's clear that the model is trying hard to preserve the pose, albeit at the cost of sharpness.

5.6.1 Relationship between IoU and I_{sim}

Following the anecdotal observation in the previous section, we decided to explore the relationship between IoU and I_{sim}. Figure 6 depicts I_{sim} as a function of IoU for (a) pose-only setting and (b) autoencoder. The two metrics have statistically significant Pearson correlation of 0.4 and 0.2 for (a) and (b) respectively. Moreover, we see that the distribution of IoU is very different between (a) and (b).

For pose-only, the distribution is bimodal. In the majority of cases, the generated image is unrecognizable to mask R-CNN and IoU=0. However, when a mask is detected, it tends to be very accurate. This is in contrast to the autoencoder setup, where IoU follows a normal distribution. This observation points to a trade-off between pose accuracy and sharpness. Our current architecture struggles to realize both at the same time.

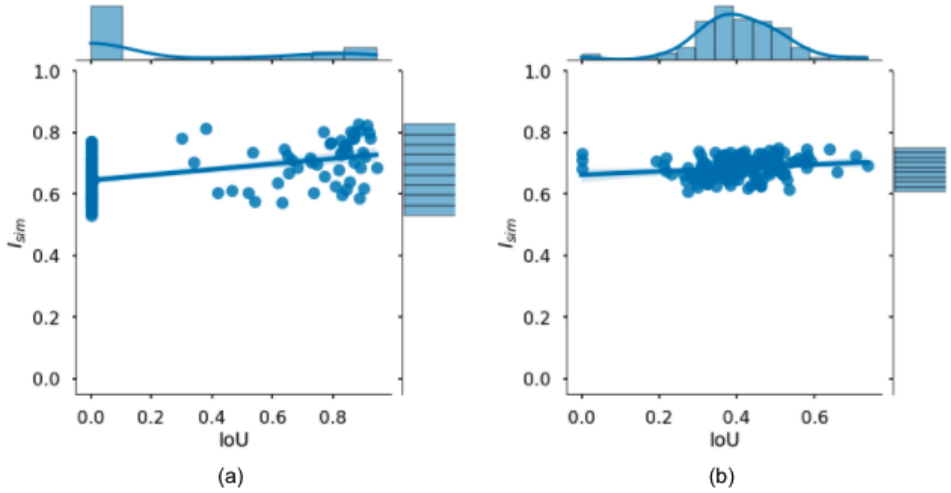


Figure 6: Relationship between I_{sim} and IoU in (a) pose-only and (b) autoencoder. The reconstruction loss used in (a) emphasizes correct prediction of the pose. This leads to a bimodal distribution of IoU values, where instance segmentation either fails (in most cases), or is very accurate. The full image reconstruction loss used by the autoencoder results in a normal distribution over IoU.

6 Conclusion

We presented a flexible approach for conditional generation of objects given any combination of three desired input types: object (e.g. car), pose, and background. This approach can help us generate realistic training data in a controllable way and at large scale. Although we see some encouraging early results, there's plenty of room for improvement. In particular, we observed that conditioning on more than one input type can result in very rapid degradation of generated image sharpness.

We partially attribute this finding to a relatively low capacity of our model. Our evidence for this hypothesis is the large performance gap we observe between oracle 1 (auto-encoder) and oracle 3. In other words, oracle 3 suggests that it is possible to generate much higher quality images.

In addition to exploring architectures with higher capacity, we hope to explore the role of the components of the loss function and the relative importance of each. Through a more systematic study of the loss function we can better understand the trade-off between sharpness and adhering to the provided inputs. Finally, we note that our methodology is agnostic to the input object type. We plan to explore this methodology for generating images of humans, animals, and inanimate objects, given desired input pose and background.

References

[1] Yuxuan Liu, Yixuan Yuan, and Ming Liu. Ground-aware monocular 3d object detection for autonomous driving. *CoRR*, abs/2102.00690, 2021.

[2] Tony Lockwood. Exploring how smart infrastructure can help ford build a great self-driving service for miamians, Mar 2021.

[3] Epic Games. Unreal engine.

[4] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[5] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[6] Aliaksandr Siarohin, Stéphane Lathuilière, Enver Sangineto, and Nicu Sebe. Appearance and pose-conditioned human image generation using deformable gans. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[7] Yifang Men, Yiming Mao, Yuning Jiang, Wei-Ying Ma, and Zhouhui Lian. Controllable person image synthesis with attribute-decomposed gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[8] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[9] Ilya Kavalerov, Wojciech Czaja, and Rama Chellappa. A multi-class hinge loss for conditional gans. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1290–1299, 2021.

[10] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[15] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[16] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[17] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[18] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6490–6499, 2019.