



[Re] Replicating and Improving GAN2Shape Through Novel Shape Priors and Training Steps

Alessio Galatolo^{1,2, } and Alfred Nilsson^{1,2, }

¹Equal Contribution – ²KTH Royal Institute of Technology, Stockholm, Sweden

Edited by

Koustuv Sinha,
Sharath Chandra Raparthy

Reviewed by

Anonymous Reviewers

Received

04 February 2022

Published

23 May 2022

DOI

10.5281/zenodo.6574685

Reproducibility Summary

Scope of Reproducibility

Pan et al. [1] propose an unsupervised method named GAN2Shape that purportedly is able to recover 3D information stored in the weights of a pre-trained StyleGAN2 model, to produce 3D shapes from 2D images. We aim to reproduce the 3D shape recovery and identify its strengths and weaknesses.

Methodology

We re-implement the method proposed by Pan et al. [1] with regards to 3D shape reconstruction, and extend their work. Our extensions include novel prior shapes and two new training techniques. While the code-base relating to GAN2Shape was largely rewritten, many external dependencies, which the original authors relied on, had to be imported. The project used 189 GPU hours in total, mostly on a single Nvidia K80, T4 or P100 GPU, and a negligible number of runs on a Nvidia V100 GPU.

Results

We replicate the results of Pan et al. [1] on a subset of the LSUN Cat, LSUN Car [2] and CelebA [3] datasets and observe varying degrees of success. We perform several experiments and illustrate the successes and shortcomings of the method. Our novel shape priors improve the 3D shape recovery in certain cases where the original shape prior was unsuitable. Our generalized training approach shows initial promise, but has to be confirmed with increased computational resources.

What was easy

The original code is easily runnable on the correct machine type (Linux operating system and CUDA 9.2 compatible GPU) for the specific datasets used by the authors.

What was difficult

Porting the model to a new dataset, problem setting or a different machine type is far from trivial. The poor cohesion of the original code makes interpretation very difficult,

Copyright © 2022 A. Galatolo and A. Nilsson, released under a Creative Commons Attribution 4.0 International license.

Correspondence should be addressed to Alessio Galatolo (galatolo@kth.se)

The authors have declared that no competing interests exist.

Code is available at <https://github.com/alessioGalatolo/GAN-2D-to-3D>. – SWH swh:1.dir:531d1456baa3bb553ce549785158be7005c682c7.

Open peer review is available at <https://openreview.net/forum?id=B8mxkTzX2RY>.

and that is why we took care to re-implement many parts of the code using the decoupling principle. The code depends on many external implementations which had to be made runnable, which caused a significant development bottleneck as we developed on Windows machines (contrary to the authors). The exact loss functions and the number of training steps were not properly reported in the original paper, which meant it had to be deduced from their code. Certain calculations required advanced knowledge of light-transport theory, which had no familiarity to us, and had to be mimicked and could not be verified.

Communication with original authors

We did not communicate with the original authors.

1 Introduction

Image generation has been a hot topic within generative models as they represent an intuitive problem whose results are easily accessible by the public. One of the models that has received a lot of public attention is StyleGAN (Karras, Laine, and Aila [4]). The network's architecture has been refined through multiple iterations in StyleGAN2 [5], StyleGAN2-ADA [6] and StyleGAN3 [7]. StyleGAN2 improves on the first version by, among other things, adding a projection method onto the latent space, which allows the inversion of an image into its latent representation.

Methods like GAN2Shape [1] aim at exploiting the information that is already stored in the generator of a pre-trained StyleGAN2 model to go beyond generating synthetic 2D images. In particular, this method aims to extract the 3D shape of the preminent object in any image. This is intuitively possible due to the size of the training dataset of the StyleGAN2 model, and its ability to generate images of an object from multiple views and lighting directions by varying \mathbf{w} . The authors of GAN2Shape use StyleGAN2 networks pre-trained on different dataset categories and five different feature extraction models to derive the shape information for images belonging to the same dataset categories. This method, compared to many others [8, 9, 10, 11], has the advantage of being completely unsupervised, and not requiring a change in the training process of the classical 2D GAN.

In this article, we describe our replication of GAN2Shape [1] and report mixed results. We perform several experiments and we illustrate the successes and shortcomings of the method. Further, we extend the method improving the original results in several cases.

2 Scope of reproducibility

The authors of GAN2Shape make the following claims:

1. Their framework does not require any kind of annotation, keypoints or assumption about the images
2. Their framework recovers 3D shape with high precision on human faces, cats, cars, buildings, etc.
3. GAN2Shape utilizes the intrinsic knowledge of 2D GANs
4. The 3D shape generated immediately allows for re-lighting and rotation of the image.

3 Methodology

Our initial intent of re-implementing the source code from the description of the paper had to be abandoned due to the lack of detailed information of some key points in the method. We, therefore, decided to follow a different approach integrating both the details from the authors' code and the paper's description. While trying to always base our implementation on the paper's description we found some parts (particularly the loss functions) that differed from the actual code and decided to follow the latter instead.

The resources we used were mainly the authors' code, the code and documentation of all the out-sourced methods the authors borrowed: StyleGAN2 [5] (code), Unsup3D [12] (code), Semseg [13] (code) and BiSeNet [14, 15] (code). The GPUs used were multiple and varied depending on availability: Nvidia Tesla K80, T4, V100 and P100.

3.1 Model descriptions

To extract the implicit 3D knowledge of pre-trained StyleGAN network, Pan et al. [1] propose an elaborate scheme involving five different neural networks. Each network models a particular quantity corresponding to the view and lighting directions, the depth of the image, and the albedo. The **View** and **Light** (V and L , resp.) networks operate in a *encoder* type manner, trying to obtain a low-dimensional vector representation of the camera view direction \mathbf{v} and the direction of light \mathbf{l} illuminating the object in the picture. The **Depth** and **Albedo** (D and A , resp.) networks utilize *auto-encoder* architectures¹ to obtain image-resolution depth maps \mathbf{d} and diffuse reflections (albedo) \mathbf{a} off the object's presumed surface.

The real GAN knowledge extraction happens in the final network, the **Offset** encoder E , combined with the pre-trained StyleGAN2 generator, G . The offset encoder aims to learn a latent representation \mathbf{w} of images with randomly sampled view and light directions, *pseudo-samples*. Paired with G , this allows the creation of new realistic samples $\tilde{\mathbf{I}}_i = G(\mathbf{w}'_i)$ with new view and lighting directions, denoted *projected samples*. The projected samples then serve as extended training data, providing multiple view-light direction variations of the original image.

To use the components \mathbf{v} , \mathbf{l} , \mathbf{d} and \mathbf{a} to obtain a reconstructed image, the authors utilize a pre-trained neural renderer developed by Kato, Ushiku, and Harada [16], which we denote by Φ .

Training Procedure – The training process of this method can be divided into 3 different steps, where the different networks involved are trained separately. In the original paper, these steps are done sequentially and for one image at a time, as shown in Figure 1, and each step is repeated multiple times before moving into the following one. The result is a model that can predict the depth map for only one image. All of the networks are trained using the Adam optimization algorithm.

Prior pre-training. Before attempting to learn the true shape of an object, the depth network is initialized by pre-training it on a fixed prior shape. For this purpose Pan et al. [1] propose to use an *ellipsoid* shape as the shape prior. We utilized this ellipsoid prior to reproduce the results of Pan et al. [1], and we extended their work by also evaluating two new different priors.

Step 1 optimizes only the A network according to Equation 1. Given an input \mathbf{I} , the first four networks predict their components \mathbf{v} , \mathbf{l} , \mathbf{d} , \mathbf{a} , and we obtain a reconstructed image $\hat{\mathbf{I}} = \Phi(\mathbf{v}, \mathbf{l}, \mathbf{d}, \mathbf{a})$ ².

$$\mathcal{L}_{\text{step1}}(\mathbf{I}, \hat{\mathbf{I}}) = \|\mathbf{I} - \hat{\mathbf{I}}\|_1 + \lambda_s \mathcal{L}_s(D(\mathbf{I})) + \lambda_p \mathcal{L}_p(\mathbf{I}, \hat{\mathbf{I}}) \quad (1)$$

Step 2 optimizes the E network according to Equation 2. Using the \mathbf{d} and \mathbf{a} components given in the last step 1 iteration, and random directions $\mathbf{v}'_i, \mathbf{l}'_i$, we generate N_p new pseudo-images \mathbf{I}'_i . For each \mathbf{I}'_i we predict $\Delta\mathbf{w}_i = E(\mathbf{I}'_i)$, which serves as input to the StyleGAN generator network G and obtain the projected images $\tilde{\mathbf{I}}_i$.

$$\mathcal{L}_{\text{step2}}(\mathbf{I}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\mathbf{I}'_i - G(\mathbf{w} + E(\mathbf{I}'_i))\|_1 + \lambda_1 \|E(\mathbf{I}'_i)\|_2 \quad (2)$$

Step 3 optimizes the L , V , D and A networks according to Equation 3. It consists in part of $\mathcal{L}_{\text{step1}}$. The second part utilizes the projected samples from the last iteration of step

¹We refer to tables 5-7 of the original paper ([1]) for the exact architectures.

² \mathcal{L}_p is a neural network trained to predict similarities between images [17] and \mathcal{L}_s is a term that encourages smoothness of the resulting depth maps (as described in [18]). We refer to our code for the weights λ_i .

2. For each projected sample $\tilde{\mathbf{v}}_i = V(\tilde{\mathbf{I}}_i)$, $\tilde{\mathbf{l}}_i = L(\tilde{\mathbf{I}}_i)$ is calculated. Combined with \mathbf{d} and \mathbf{a} from the original image, they can be used to reconstruct each projected sample from the components $\bar{\mathbf{I}} = \Phi(\tilde{\mathbf{v}}_i, \tilde{\mathbf{l}}_i, \mathbf{d}, \mathbf{a})$.

$$\mathcal{L}_{\text{step3}}(\mathbf{I}, \bar{\mathbf{I}}) = \frac{1}{N_p} \sum_{i=1}^{N_p} [\mathcal{L}_p(\mathbf{I}, \bar{\mathbf{I}}_i) + \|\mathbf{I} - \bar{\mathbf{I}}_i\|_1] + \mathcal{L}_{\text{step1}}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_2 \mathcal{L}_s(\mathbf{D}(\mathbf{I})) \quad (3)$$

Stages. The steps are repeated for a number of *stages*. In each, the steps are trained for a different number of iterations (see Table 1 in subsection 5.5 in the appendix for details).

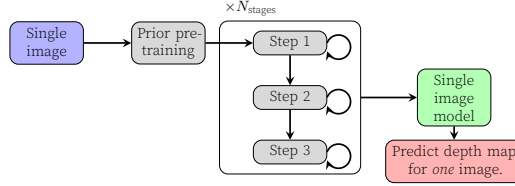


Figure 1. Schematic of the original training process.

Novel Shape Priors – The first novel prior we consider is a masked box. Using the mask returned by the parsing model developed by Zhao et al. [19] we extrude the relevant object from the background, in a step-like manner. Improving on this idea, we also smooth the transition from the object to the background. This is done by using three 2D convolutions, where we convolve the masked box shape with a 11×11 filter of ones. Renormalizing the convolved shape, we obtain Figure 2c denoted as ‘smoothed box’.

The last prior we tested is obtained by normalizing the score (or “confidence”) that the parsing model gives to each pixel. We use this confidence to project the object, i.e. a pixel that is within the category with more confidence will be farther projected. This prior is similarly smoothed by convolutions and is denoted as ‘confidence based’.

Figure 2 shows a visual representation of the prior shapes used for an example image taken from the CelebA dataset.

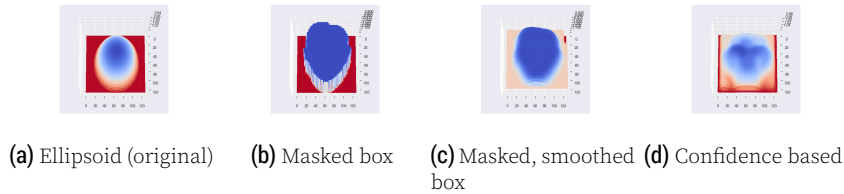


Figure 2. Original vs. our novel shape priors, shown on the CelebA (face) dataset

3.2 Generalized Training Procedure

Given the single-use nature of the model obtainable with the original training procedure, we decided to develop an alternative training procedure to favor a general model M^* usable for all images belonging to the same distribution as the training dataset \mathcal{D} . We propose to pre-train the depth net D on all images first, instead of repeating the process for each image. We also modify Step 1, 2 and 3 by greatly lessening the number of iterations given to a single image and breaking up the sequential training of the original method into a few iterations per example, and instead introducing N_e epochs and batch training to compensate, increasing resource utilization and training speed. To facilitate

understanding of our modifications to the training procedure, we provide a schematic in Figure 3. It can be compared to the original shown in Figure 1. Let us note that the

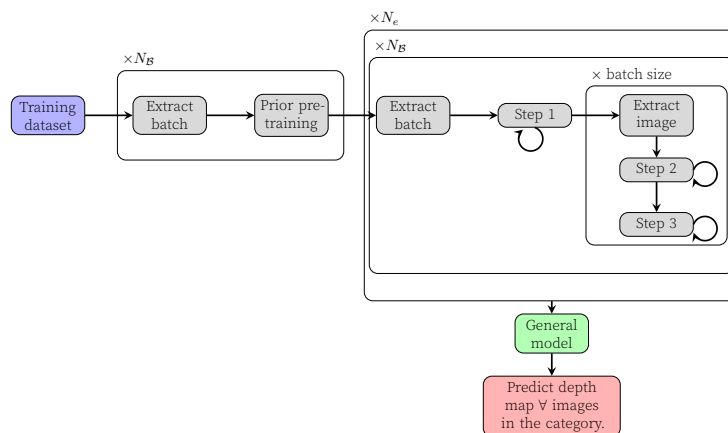


Figure 3. Schematic of our new training process designed to favor generalization.

original authors also briefly mention a ‘joint training’ that should improve the generalization ability of the model, however, its performance is not properly reported and it only represents a mini-batch extension of the pre-training step.

3.3 Datasets

We aimed to reproduce the authors’ results on the LSUN Car, LSUN Cat [2] and CelebA [3]. From these datasets, the authors selected a subset consisting of 10 images of cars, 216 images of cat faces, and 399 celebrity faces. Like the authors, we used RGB images of three color channels, resized to 128×128 pixel resolution. No further preprocessing was applied.

3.4 Hyperparameters

For replication purposes, the original hyperparameters by Pan et al. [1] were used, but we also tried tuning some parameters that we believe are key to the method: the number of projected samples, N_p , for each image and the number of epochs for pre-training the depth network. N_p was varied within $\{2, 4, 8, 16, 32\}$. In our tests we found the values 4, 8 and 8, respectively for the LSUN Car, LSUN Cat and CelebA dataset, to be the threshold after which the improvements in image quality start greatly decreasing (see subsection 5.11 in Appendix for more details).

The number of epochs for the depth network pre-training was varied within $\{100, 500, 1000, 2000\}$. This pre-training affects how irregular the depth map predictions are. We believe that using a threshold for the loss to check the convergence would be preferable as the number of epochs selected by the authors (1000) is enough in most cases but not in all. We attribute irregularity in some of our results to this issue.

3.5 Experimental setup and code

For each dataset we run our implementation of the framework from Pan et al. [1] on the images that were selected by the authors, the procedure saves a checkpoint for each network. These checkpoints are later fed the original image to get the generated result. The evaluation of the results was only qualitative as all the datasets we explored do not have a ground truth for comparison. We instead relied on a manual evaluation.

Our code is available at <https://github.com/alessioGalatolo/GAN-2D-to-3D>. Our results are available interactively under the docs folder and at alessiogalatolo.github.io/GAN-2D-to-3D/.

3.6 Computational requirements

Most of the experiments we ran were on a Intel(R) Xeon(R) CPU @ 2.20GHz with 2 cores available and a Nvidia Tesla P100-PCIE-16GB. Since the framework described by Pan et al. [1] is instance-specific, we report the average time for completing the projection of a single image: 96m and 28s for an image in the CelebA dataset, 95m and 43s for a LSUN Cat image and 74m and 32s for a LSUN Car image.

4 Results

The model correctly learned the shape and the texture of many images, although some examples were less successful than others. For example, the model converged to believable shapes for two of the cars in Figure 4, but the shape of the right-most car is debatable.

In the following sections we show the reconstructed depth map and 3D projection of some images chosen as representative of the dataset. All of the images that follow have the background cut from the actual object, this was only done for ease of illustration and was not done for the actual training process since the original authors do not mask the background in all cases. It is also difficult to illustrate the results fairly in 2D images, so we invite the reader to visit our website with *interactive* 3D plots³.

4.1 Results reproducing the original paper

LSUN Car – We present the results on LSUN Car dataset in Figure 4. Most features are projected in the correct direction and show details that are correctly outward projected from the main object. This result supports all the claims made in section 2 as we did not use any annotation or assumption for the images, many details were retrieved with high precision using the StyleGAN knowledge and we were able to easily make a rotation of the image (see interactive web-page).

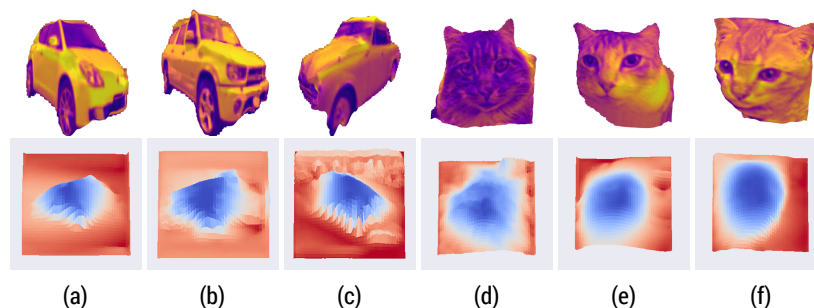


Figure 4. LSUN Car and Cat

LSUN Cat – The second experiment was conducted on the LSUN Cat dataset. The results were slightly poorer compared to the LSUN Car dataset. The face of the cats gets properly recognized, but some details like the nose are not protruded from the rest of the face and are generally on the same plane, see Figure 4. Some images present some irregularities in the form of spikes and hills (d). The rotation (f) does not result in a completely natural

³alessiogalatolo.github.io/GAN-2D-to-3D/

image as part of the face of the cat appears on the same plane. This experiment does not support claims 2 and 4 in some cases (e.g. figures 4 (d) and (f) negate claims 2 and 4 respectively) while it does for claims 1 and 3 (section 2).

CelebA – The third experiment conducted on the CelebA dataset shows that most of the face are correctly portrayed with the only exception of the border of the face e.g. chin and forehead that sometimes is not included in the projection, see Figure 5 (b). Also we found out that the method does not behave well with faces that are viewed from the side, see Figure 5 (c), where the face still gets a projection as it was viewed from the front. As a consequence of this, the rotation of side faces does not result in a good image. This experiment supports claims 1-4 (section 2) only for some faces and claims 1 and 3 for those viewed from the side.

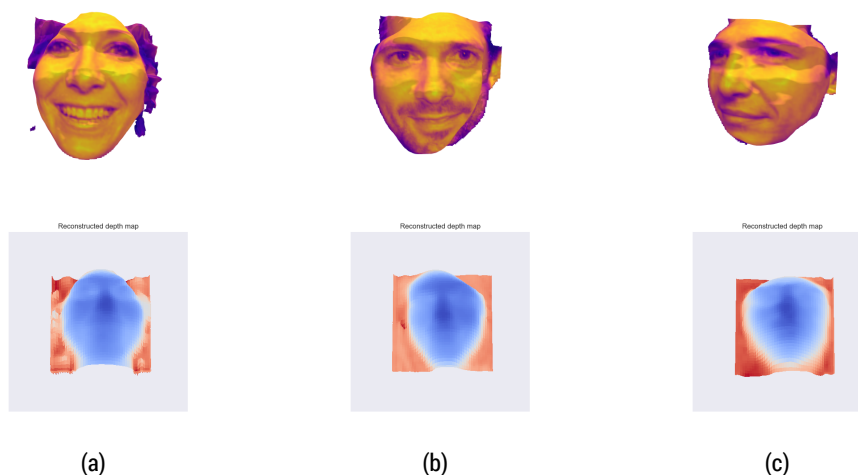


Figure 5. CelebA

4.2 Results beyond the original paper

The effects of shape priors – The original paper did not specify the exact reasons for choosing an ellipsoid prior for the pre-training of the depth net, therefore we decided to experiment with multiple prior shapes as well as no prior shape.

No prior. With the goal of assessing the results of this method when no prior shape is given, we ran a test on one image from the LSUN Car dataset without any prior pre-training, and with random initialization. The reconstruction objective is still satisfied very well, but it has converged to an extremely noisy depth map (see Figure 9 in subsection 5.6 in the appendix). This briefly shows that this method would not work without a strong shape prior to guide it towards a reasonable shape.

Smoothed Box Prior. The first extension experiment was done by testing the first of the prior shapes we proposed, the smoothed box prior. Figure 6 shows the smoothed box prior tested on the LSUN Cat and CelebA dataset where it can be seen how it is better at understanding the structure of the nose and face in general.

Confidence-Based Prior. Another experiment we performed focused on the performance of the second prior we presented, the confidence based prior. Figure 7 shows some results on the datasets considered in this paper. The results are most promising in the CelebA dataset where the image of a face is correctly projected even if viewed from the side.

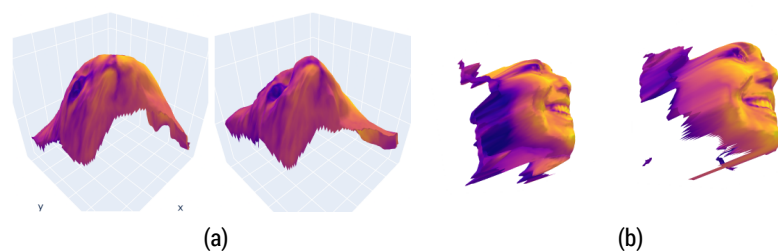


Figure 6. Example result for two different image examples from the LSUN Cat and CelebA datasets. For each example, the left-most figure corresponds to the ellipsoid and right-most figure corresponds to the smoothed masked box prior.



Figure 7. Results with the confidence based prior.

Generalized Training Procedure – We demonstrate the results of our new training loop on LSUN Cat. We note again that the difference to the previous demonstration on LSUN Cat, is that a single network D^* was used to predict all of the images, as opposed to a different network D_i for each image I_i . The general model was trained on a limited subset of 30 images from LSUN Cat. It was trained for a modest 60 epochs which results in approximately 60% of the weight updates per image of the original method. Figure 8 shows the projection of some images from the LSUN Cat dataset. One can observe that the method recognizes the general structure of the cat’s face but also presents some artefacts in some specific parts of the face e.g. the second cat’s cheek is further projected than where it should and similarly for the third cat’s chin.

Improved initialization – Our final experiment is inspired by the observation of the dependency of the method to the number of pseudo-samples N_p , and the variability that follows in the results depending on their quality, as discussed in subsection 5.3.1. We experiment with drastically increasing this number from 16 to 128 for 10 short epochs, in which each training step is performed only once. We observe marginal improvements in the predicted shape (Figure 8) and larger improvements in the smaller details/features. See the subsection 5.10 in the appendix for further detail.

Training step 1 was not changed and it is allowed to converge in the first stage, as it does not involve the projected samples. See Table 2 in the appendix for an exact description of the number of iterations. All other parameters were left as in subsection 4.2.1, with the smoothed box prior. We experimented with two of the worst performers from the LSUN Cat dataset to evaluate whether this method could improve the results, see Figure 16. We applied the same idea to the general model described in sections 3.2, 4.2.2 and saw improvements, see Figure 8. The results can be compared to Figure 14.

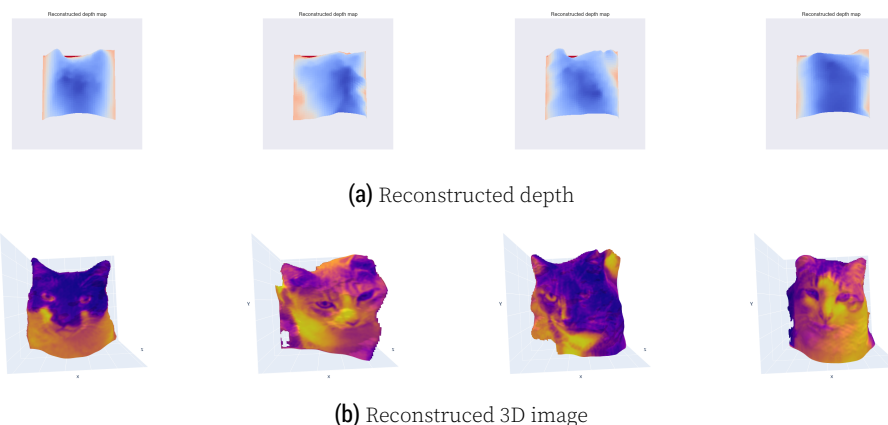


Figure 8. Depth map predictions for a few image samples from the training set $\mathcal{D} \subset$ LSUN Cat dataset, all using one and the same **general** model M^* trained with **initialization** iterations.

5 Discussion

5.1 What was easy

The authors provide a clear specification of the Python package dependencies, as well as other dependencies. Additionally, they provide scripts for easy downloading of a select few datasets and pre-trained model weights. They precisely state how to execute the training script and how to run the model for evaluation. Note that this refers to running the original code and that modifying and extending the code brought many difficulties, as explained in the next section.

5.2 What was difficult

The paper by Pan et al. [1] did not contain enough information for a successful reimplementation. Many details had to be discerned or guessed from their code. Furthermore, the quality of said code does not allow for a quick interpretation. For example, deducing the training loop and the number of iterations for each step was further complicated by the poor cohesion of the original code: the trainer script was heavily mingled with model class, using class members of the model object to increment training steps and nested function calls back and forth between the trainer and model classes.

The components \mathbf{v} , \mathbf{l} , \mathbf{d} and \mathbf{a} were not enough to pass in to the neural renderer to reconstruct an image. In reality, several calculations of quantities such as diffuse shading and texture were needed to be fed into the neural renderer, using concepts from light transport theory that were not mentioned in the paper.

Another difficulty was the heavy reliance on external pre-trained neural networks. The neural renderer [16], in particular, posed several problems. The major one was incompatibility with Windows machines. To be able to develop on our personal machines, we had to make manual edits of the neural renderer script and different CUDA files.

Another challenge with this method is the lack of objective quantitative metrics to evaluate the success of the models. One instead has to rely almost entirely on qualitatively gauging the shape reconstructions by eye.

5.3 Conclusions

Variability of the results – We observed that the method is very sensitive to various random factors and identical runs may yield different results, see Figure 12. One factor may be the random initialization of the networks, but we do not believe it is the dominating factor, since the depth network is pre-trained on a fixed prior shape each run. Rather, as mentioned by the authors [1], the quality of the projected samples varies. Additionally, we only sample 8 – 16 different view-light directions in each step 2 iteration, which may be too few projected samples for a robust model. Since this sampling is random, increasing the number of samples should assure the inclusion of meaningful view and light projections (experimental backing in subsection 5.7 in the appendix).

Catastrophic forgetting – We have observed that the instance-specific model forgets the previous training images (see subsection 5.8 in the appendix, Figure 13), and thus has no generalization capability. This is not necessarily a problem if one has time and computational resources. It can also be argued that this is exactly what is intended with this model, and that generalization is up to the training dataset of the StyleGAN model. It does, however, limit the usefulness of the model. As an example, the training time for one 128×128 pixel RGB image using a Tesla K80 GPU was about 2.5 hours, which seems exceedingly costly for just one low-resolution depth map. We argue that a *general* model would have more use. The ideal scenario would be a model D^* trained on \mathcal{D} that is able to accurately predict $\mathbf{d}_i = D^*(\mathbf{I}_i) \forall \mathbf{I}_i \in \mathcal{D}$, and even extend to unseen testing data belonging to the same distribution as \mathcal{D} . This discussion is what urged us to explore the altered training procedure of sections 3.2 and 4.2.2.

Final conclusions – We were able to replicate some of the results of Pan et al. [1] on the datasets LSUN Car, LSUN Cat and CelebA. We identified several failure modes and limitations of the model, and back it up with experimental evidence. Examples are the variability and sensitivity to the projected samples, the heavy dependence on shape priors and the computational costliness of the single-use model - all of which were not adequately accounted for in the original paper.

We propose a new prior shape, the smoothed box prior, that has shown very promising results especially for fine details and complex object structures. We propose a second prior shape, confidence-based, that has shown best results in the face dataset. We finally suggest two new training procedures that produce better results and are better at generalizing than the original model by Pan et al. [1].

We recognize the limitations of this work as we were only able (due to the restricted computational power) to test the method on part of the dataset. For example, the Cat’s dataset used by the authors contains more than 200 images but we were able to only test few of them. We speculate that some images in the dataset could yield better results than those reported here. However, we believe that few bad projected images should be enough to claim the ineffectiveness of the method at least in some particular cases.

Another limitation of our work is the lack of quantitative evaluation methods. The original authors propose their results also on the BFM benchmark [20] where it is possible to use some metrics to accurately evaluate the results.

5.4 Future work

We speculate that it would be interesting to adapt the same method to StyleGAN3 [7] where the network has been modified to support training with fewer samples, leaving the question if the network still retains enough information that is needed for GAN2Shape to work. Future work could also explore the use of our priors on datasets where the original method failed (e.g. the LSUN Horse dataset). We speculate that, since our prior

captures the boundaries of the object very well (compared to the ellipsoid where the boundaries are only used to position the origin), it could achieve better results in complex 3D objects where the shape cannot be simplified into an ellipse. A limitation of this method is that it does not use voxels, but learns a height map. This disallows realistic shape reconstructions and more complex geometries with multiple x and y values for each z value etc. Future work should investigate whether this model could be extended to predict voxels instead of height maps. Given our promising results with the generalizing trainer, which was obtained through only a few epochs of training, we believe that it should be further explored with increased epochs and training set size.

References

1. X. Pan, B. Dai, Z. Liu, C. C. Loy, and P. Luo. "Do 2D GANs Know 3D Shape? Unsupervised 3D shape reconstruction from 2D Image GANs." In: (2021). arXiv:2011.00844 [cs.CV].
2. F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop." In: *arXiv preprint arXiv:1506.03365* (2015).
3. Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep Learning Face Attributes in the Wild." In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.
4. T. Karras, S. Laine, and T. Aila. "A style-based generator architecture for generative adversarial networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.
5. T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. "Analyzing and improving the image quality of stylegan." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8110–8119.
6. T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. "Training generative adversarial networks with limited data." In: *arXiv preprint arXiv:2006.06676* (2020).
7. T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. "Alias-Free Generative Adversarial Networks." In: *Proc. NeurIPS*. 2021.
8. S. Lunz, Y. Li, A. Fitzgibbon, and N. Kushman. "Inverse graphics gan: Learning to generate 3d shapes from unstructured 2d data." In: *arXiv preprint arXiv:2002.12674* (2020).
9. P. Henzler, N. J. Mitra, and T. Ritschel. "Escaping plato's cave: 3d shape from adversarial rendering." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9984–9993.
10. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. "3d shapenets: A deep representation for volumetric shapes." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.
11. H. Wang, J. Yang, W. Liang, and X. Tong. "Deep single-view 3D object reconstruction with visual hull embedding." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8941–8948.
12. S. Wu, C. Rupprecht, and A. Vedaldi. "Unsupervised learning of probably symmetric deformable 3d objects from images in the wild." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1–10.
13. H. Zhao. *semseg*. <https://github.com/hszhao/semseg>. 2019.
14. C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. "Bisenet: Bilateral segmentation network for real-time semantic segmentation." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 325–341.
15. C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang. "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation." In: *International Journal of Computer Vision* 129.11 (2021), pp. 3051–3068.
16. H. Kato, Y. Ushiku, and T. Harada. "Neural 3D Mesh Renderer." In: *CoRR* abs/1711.07566 (2017). arXiv:1711.07566. URL: <http://arxiv.org/abs/1711.07566>.
17. J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution." In: (2016). arXiv:1603.08155 [cs.CV].
18. T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. "Unsupervised Learning of Depth and Ego-Motion from Video." In: *CoRR* abs/1704.07813 (2017). arXiv:1704.07813. URL: <http://arxiv.org/abs/1704.07813>.
19. H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. "Pyramid scene parsing network." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.
20. P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. "A 3D face model for pose and illumination invariant face recognition." In: *2009 sixth IEEE international conference on advanced video and signal based surveillance*. IEEE. 2009, pp. 296–301.

Appendix

5.5 Hyperparameters

Stage	Iterations/step
1	[700, 700, 600]
2, 3, 4	[200, 500, 400]

Table 1. Specification of the different stages for the single-image model.

Stage	Iterations/step	N_p
0	[700, 0, 0]	16
1-10	[1, 1, 1]	128
11	[1, 700, 600]	16
12, 13, 14	[200, 500, 400]	16

Table 2. Specification of the different stages for the single-image model with **initialization iterations**

Epochs	Iterations/step	N_p
60	[13, 22, 18]	16

Table 3. Specification of the iterations/step for the generalized model.

Epochs	Iterations/step	N_p
10	[13, 1, 1]	128
60	[13, 22, 18]	16

Table 4. Specification of the iterations/step for the generalized model with **initialization iterations**

Table 5. Hyperparameters for the general model with initialization iterations on the LSUN Cat dataset.

Parameter	Value
n_epochs_prior	1000
n_epochs_generalized	70
n_epochs_init	10
n_init_iterations	8
batch_size	10
channel_multiplier	1
image_size	128
z_dim	512
root_path	data/cat
learning_rate	0.0001
view_scale	1
refinement_iterations	1
n_proj_samples	16
rot_center_depth	1.0
fov	10
tex_cube_size	2

We refer to our GitHub repository for a complete declaration of all hyperparameters for all datasets.

5.6 Effects of shape priors

Figure 9 shows the effects of random initialization of the depth network. Figure 10 shows

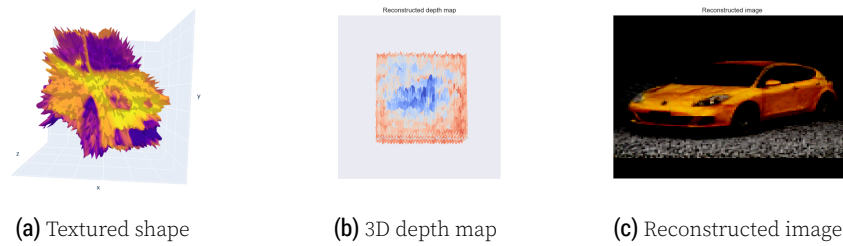


Figure 9. Results with no shape prior.

the results on the first car where it can be observed that our prior is even better than the ellipsoid at capturing fine details such as the side mirror.

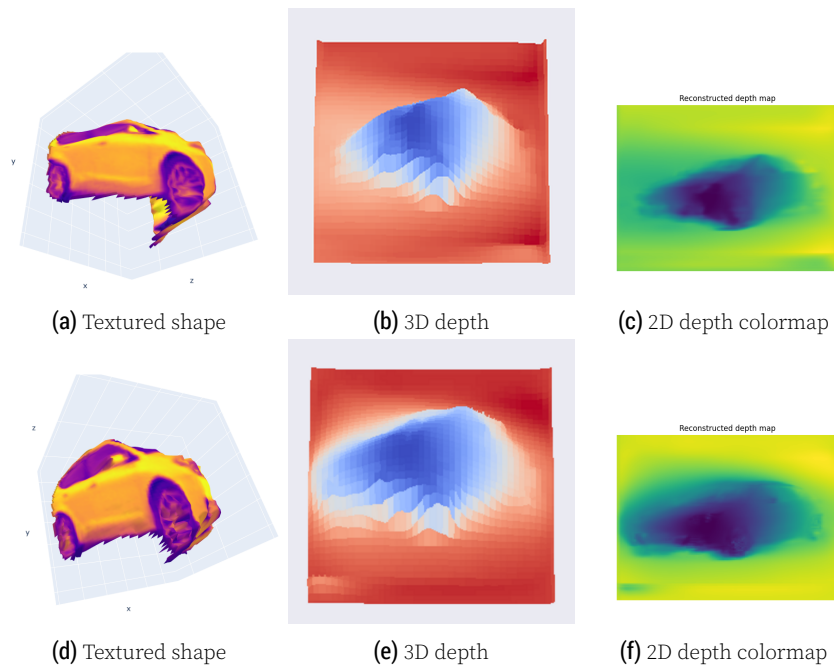


Figure 10. Ellipsoid prior (top row) vs. the smoothed masked box (bottom row) prior.

5.7 Variability of identical runs

Figure 12 shows several runs with the same configuration on the first car image ending up with different results.

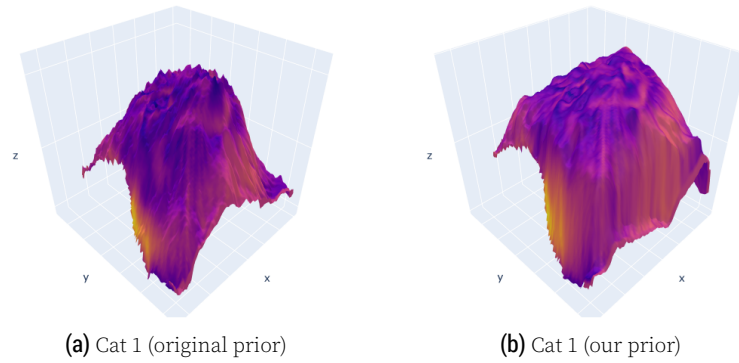


Figure 11. Results for a few other images from the LSUN Cat dataset, for the ellipsoid (left) and smoothed masked box (right) priors.

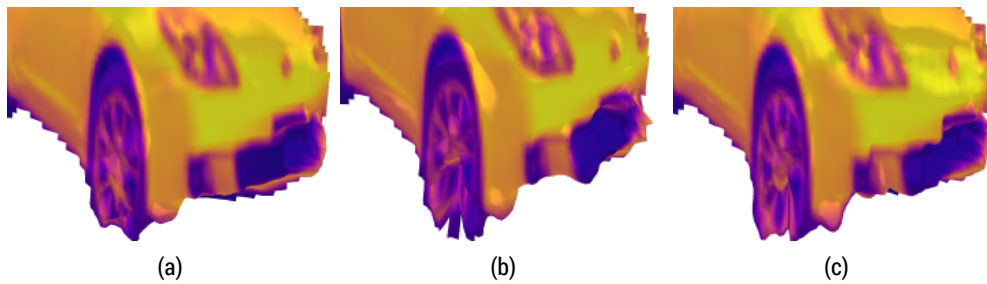


Figure 12. Several runs with identical configuration.

5.8 Catastrophic forgetting

When the training process is complete for one image \mathbf{I}_t we have confirmed that the model $M_t = \{V, L, D, A\}_t$ is able to construct a believable depth map (subsection 4.1). However, when training continues to the next image \mathbf{I}_{t+1} and M_{t+1} is obtained, we have observed that the ability to predict the depth map of the previous image deteriorates, and the problem gets worse with an increasing time discrepancy between the model and image. In other words, the depth network D_t at training step t is only usable for predicting the depth map $\mathbf{d}_t = D_t(\mathbf{I}_t)$ and so suffers from catastrophic forgetting of the previous images. This is illustrated in Figure 13.

The training time for one 128×128 pixel RGB image using a Tesla K80 GPU was about 2.5 hours, which seems exceedingly costly for just one low-resolution depth map.

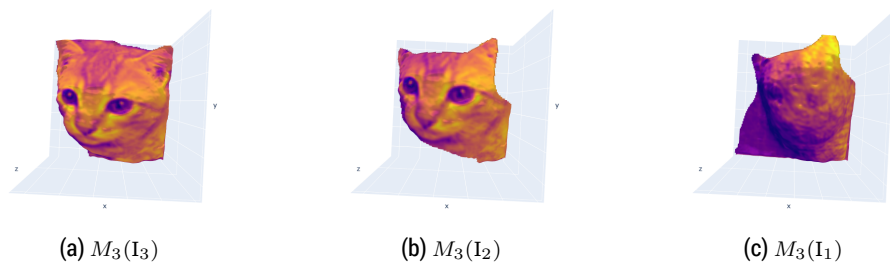


Figure 13. Depth map predictions for a few image samples from the LSUN Car dataset, illustrating catastrophic forgetting for the model M .

5.9 Additional generalized training results

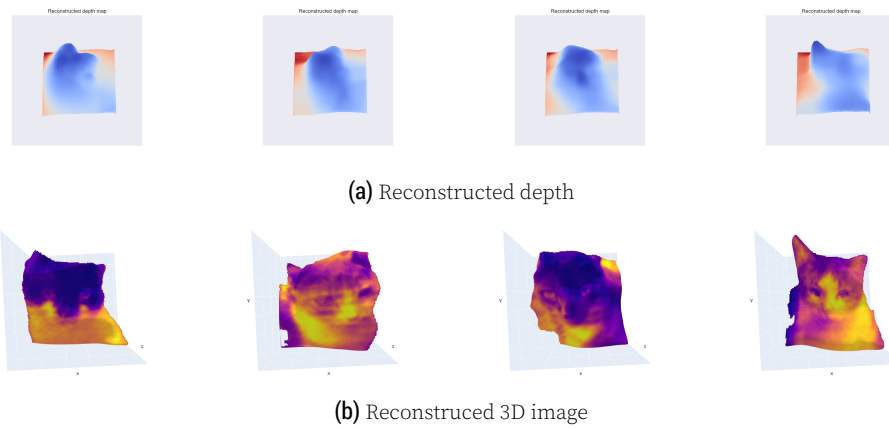


Figure 14. Depth map predictions for a few image samples from the training set $\mathcal{D} \subset$ LSUN Cat dataset, all using one and the same **general** model M^* .

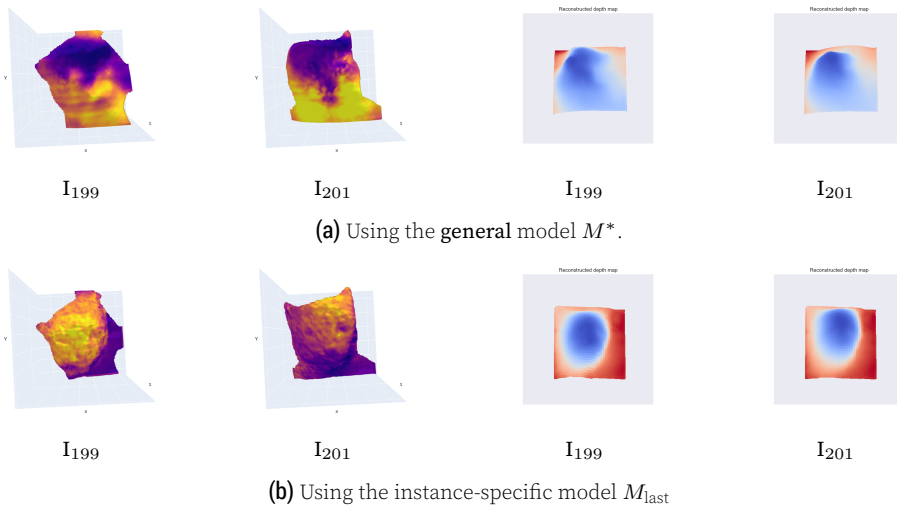


Figure 15. Depth map predictions for **unseen** image samples $\{I_{199}, I_{201}\} \notin \mathcal{D}$ from the LSUN Cat dataset.

5.10 Initialization iteration results

The observations of sections 5.3.1 and 3.4 can be condensed into two main points to form a hypothesis. Please note that our limited computational resources meant that we could not perform rigorous experimentation to confirm these observations with a large number of samples, and that this section should be viewed as a speculative experiment.

- The initial few training iterations can be viewed as an *initialization* of the weights, which depends on what projected samples are generated by the StyleGAN2 model.
- The “features” (i.e. peaks and valleys) of the depth map predictions do not qualitatively change with increasing iterations, but remain fixed except in size (i.e. the height of the peaks).

If one accepts these claims, then it is clear that the first few iterations determine the success of the shape reconstruction. That is why we experiment with drastically increasing the number of pseudo-samples during the first iterations. This reduces the bias of the initialization and reduces the relative impact that a poor projected sample generated by the GAN has on the model weights. Specifically, we increase the number of projected samples N_p from 16 to 128 for 10 short epochs, in which each training step is performed only once.

Ideally, one would of course permanently increase N_p , but with extreme costs in terms of training time. This method only added ~ 4 minutes of training time using a Tesla T4 GPU.

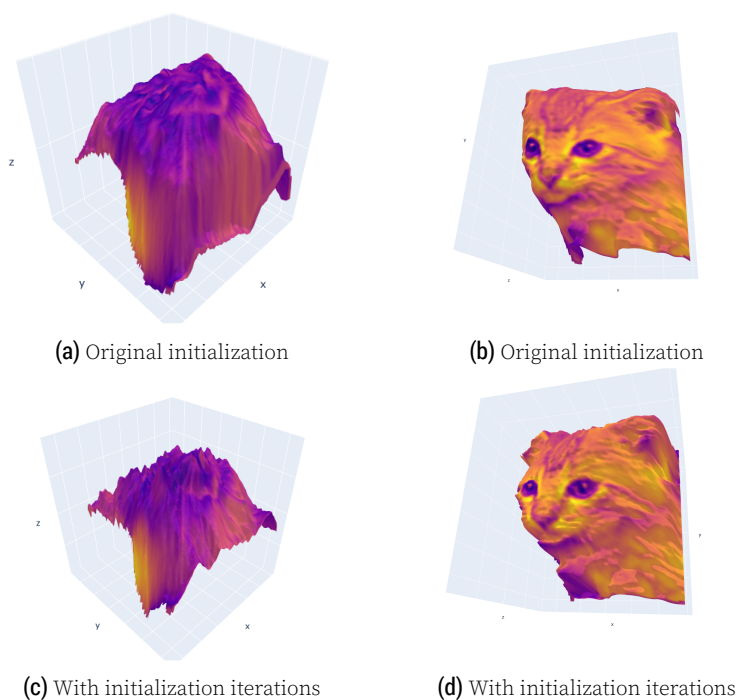


Figure 16. Results for the worst performers for the **single-image** model using the smoothed box prior, from the LSUN Cat dataset. Original initialization (top row) and using **initialization iterations** (bottom row). The leftmost cat saw the most drastic changes. While the result is a “spikey” depth map, we argue that the general shape has a better resemblance to a cat, and less square box-like as with the original initialization. The rightmost cat saw some improvement in some details such as the ears and the mouth region.

5.11 Hyperparameter tuning

We found that N_p correlates with the quality of the predicted shapes. The trend tends to be that more is better, but with diminishing returns. The biggest benefit that a large N_p has, is that strange artefacts are less likely to persist. It is difficult to pinpoint an acceptable threshold for N_p , as it varies between datasets and even between images. Therefore we believe a good compromise is to perform a few initialization iterations as described in section 4.2.3 with a large N_p (i.e. 128) and then continue training with a lower number according to the aforementioned thresholds.

To illustrate the results when varying on the number of projected samples N_{proj} we present the results on the LSUN car and Celeba dataset. In Figure 18 the first two cars (corresponding to a low N_p) have more irregular surfaces and one has a large spike, while the third is more regular. The same is observed for the Celeba faces in Figure 17, where the first face (corresponding to a low N_p) has significant irregularities across the face. As described in subsection 5.3.1, we attribute this phenomenon to lower relative impact that sampling poor view-light projections has, the larger N_p is.



Figure 17. Face 1 when trained with 4, 8, 16 and 32 (from left to right) number of projected samples.



Figure 18. Car 1 when trained with 2, 4 and 8 (from left to right) number of projected samples.