GeLaTO: Generative Latent Textured Objects

Ricardo Martin-Brualla[1(✉)], Rohit Pandey[2], Sofien Bouaziz[2],
Matthew Brown[1], and Dan B. Goldman[1]

[1] Google Research, Seattle, USA
rmbrualla@google.com, mtbr@google.com, dgo@google.com
[2] Google Research, Mountain View, CA, USA
rohitpandey@google.com, sofien@google.com

Abstract. Accurate modeling of 3D objects exhibiting transparency, reflections and thin structures is an extremely challenging problem. Inspired by billboards and geometric proxies used in computer graphics, this paper proposes Generative Latent Textured Objects (GeLaTO), a compact representation that combines a set of coarse shape proxies defining low frequency geometry with learned neural textures, to encode both medium and fine scale geometry as well as view-dependent appearance. To generate the proxies' textures, we learn a joint latent space allowing category-level appearance and geometry interpolation. The proxies are independently rasterized with their corresponding neural texture and composited using a U-Net, which generates an output photorealistic image including an alpha map. We demonstrate the effectiveness of our approach by reconstructing complex objects from a sparse set of views. We show results on a dataset of real images of eyeglasses frames, which are particularly challenging to reconstruct using classical methods. We also demonstrate that these coarse proxies can be handcrafted when the underlying object geometry is easy to model, like eyeglasses, or generated using a neural network for more complex categories, such as cars.

Keywords: 3D modeling · 3D reconstruction · Generative modeling

1 Introduction

Recent research in category-level view and shape interpolation has largely focused on generative methods [20] due to their ability to generate realistic and high resolution images. To close the gap between generative models and 3D reconstruction approaches, we present a method that embeds a generative model in a compact 3D representation based on textured-mapped proxies.

Texture-mapped proxies have been used as a substitute for complex geometry since the early days of computer graphics. Because manipulating and rendering

(a) Traditional billboards   (b) Planar proxies   (c) Free-form proxies
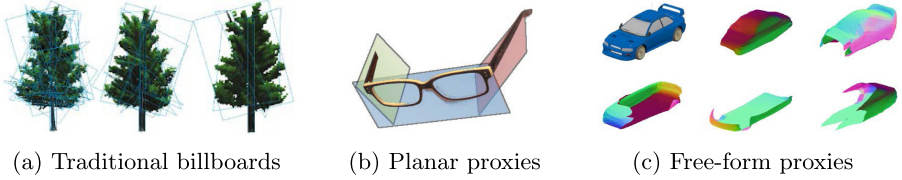
Fig. 1. Inspired by (a) traditional computer graphics billboards [12], our representation uses (b) planar proxies for classes with well-bounded geometric variations like eyeglasses, and (c) free-form 3D patches for generic classes like cars.

geometric proxies is much less computationally intensive than corresponding detailed geometry, this representation has been especially useful to represent objects with highly complex appearance such as clouds, trees, and grass [10,36]. Even today, with the availability of powerful graphics processing units, real-time game engines offer geometric representations with multiple levels of detail that can be swapped in and out with distance, using texture maps to supplant geometry at lower levels of detail.

This concept can be adapted to deep learning, for which the capacity of a network that can learn complex geometry might be larger than the capacity needed to learn its surface appearance under multiple viewpoints. Inspired by texture-mapped proxies, we propose a representation consisting of four parts: ① a 3D proxy geometry that coarsely approximates the object geometry; ② a view-dependent deep texture encoding the object's surface light field, including view-dependent effects like specular reflections, and geometry that lies away from the proxy surface; ③ a generative model for these deep textures that can be used to smoothly interpolate between models, or to reconstruct unseen object instances within the category; ④ a U-Net to re-render and composite all the Neural Proxies into a final RGB image and a transparency mask.

To evaluate our approach we capture a dataset of 85 eyeglasses frames and demonstrate that our compact representation is able to generate realistic reconstructions even for these complex objects featuring transparencies, reflections and thin features. In particular, we use three planar proxies to model eyeglasses and show that using our generative model, we can reconstruct an instance with more accuracy and $3\times$ fewer input views compared to a model optimized exclusively for that instance. We also show compelling interpolations between instances of the dataset, and a prototype virtual try-on system for eyeglasses. Finally, we qualitatively evaluate our representation on cars from the ShapeNet dataset [7], for which we use five free-form parameterized textured mesh proxies learnt to model car shapes [15].

To summarize, our main contributions are: ① a novel compact representation to capture the appearance and geometry of complex real world objects; ② a re-rendering and compositing step that can handle transparent objects; ③ a learned latent space allowing category-level interpolation; ④ few-shot reconstruction, using a network pre-trained on a corpus of the corresponding object category.

2 Related Work

2.1 3D Reconstruction

Early work in 3D reconstruction attempted to model a single object instance or static scene [34] by refining multiview image correspondences [13] along with robust estimation of camera geometry. These methods work well for rigid, textured scenes but are limited by assumptions of Lambertian reflectance. Later work attempts to address this, for example using active illumination to capture reflectance [44], known backgrounds to reason about transparency [38], or special markers on the scanner to recognise mirrors [45]. Thin structures present special challenges, which Liu et al. [25] address by fusing of RGBD observations over multiple views. Even with such specifically engineered solutions, reconstruction of thin structures, reflection and transparency remain open research problems, and strong object or scene priors are desirable to enable accurate 3D reconstruction.

Recent progress in deep learning has renewed efforts to develop scene priors and object category models. Kar et al. [19] learn a linear shape basis for 3D keypoints for each category, using a variant of NRSfM [6]. Kanazawa et al. [18] learn category models using a fixed deformable mesh, with a silhouette based loss function trained via a differentiable mesh renderer. Later work to regress mesh coordinates directly from the image, trained via cycle consistency, showed generalization across deformations for a class-specific mesh [23]. Chen et al. represent view dependent effects by learning surface lightfields [8]. Implicit surface models [9,28,32] use a fully connected network to represent the signed surface distance as a function of 3D coordinate.

2.2 Neural Rendering

Neural rendering techniques relax the requirement to produce a fully specified physical model of the object or scene, generating instead an intermediate representation that requires a neural network to render. We refer the reader to the comprehensive survey of Tewari et al. [41]. Recent works use volumetric representations that can be learned on a voxel grid [27,39], or modeled directly as a function taking 3D coordinates as input [30,40]. These methods tend to be computationally expensive and have limited real-time performance (except for [27]). Neural textures [43] jointly learn features on a texture map along with a U-Net. IGNOR [42] incorporates view dependent effects by modelling the difference between true appearance and a diffuse reprojection. Such effects are difficult to predict given the scene knowledge, so GAN based loss functions are often used to render realistic output. Deep Appearance Models [26] use a conditional variational autoencoder to generate view-dependent texture maps of faces. Image-to-image translation (pix2pix) [16] is often used as a general baseline. HoloGAN learns a 3D object representation such that sampled reprojections under a transform fool a discriminator [31]. Point-cloud representations are also popular for neural rerendering [29,33] or to optimize neural features on the point cloud itself [2].

3 Generative Latent Textured Objects

Our representation is inspired by proxy geometry used in computer graphics. We encode the geometric structure using a set of coarse proxy surfaces shown in Fig. 1, and shape, albedo, and view dependent effects using view-dependent neural textures. The neural textures are parameterized using a generative model that can produce a variety of shape and appearances.
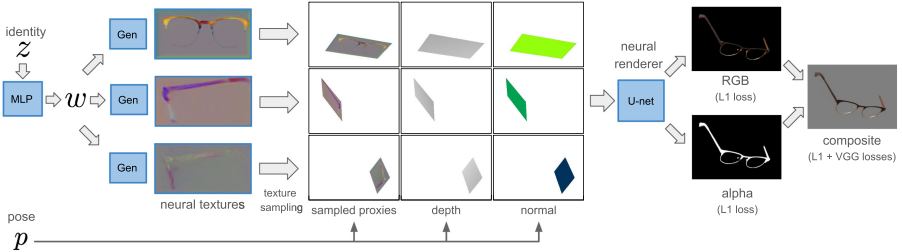

Fig. 2. Network architecture. See Sect. 3.2 for details.

3.1 Model

Given a collection of objects of a particular class, we define a latent code for each instance i as $\mathbf{z}_i \in \mathbb{R}^n$. We assume that a coarse geometry consisting of a set of K proxies $\{P_{i,1}, \ldots, P_{i,K}\}$, i.e. triangular meshes with UV-coordinates, is available. Our network computes a neural texture $T_{i,j} = \text{Gen}_j(\mathbf{w}_i)$ for each instance and proxy, where $\mathbf{w}_i = \text{MLP}(\mathbf{z}_i)$ is a non-linear reparametrization of the latent code $\mathbf{z}_i$ using an MLP. The image generators $\text{Gen}_j(\cdot)$ are decoders, that take a latent code as input and generate a feature map. To render an output view, we rasterize a deferred shading deep buffer from each proxy consisting of the depth, normal and UV coordinates. We then sample the corresponding neural texture using the deep buffer UV coordinates for each proxy. The deep buffers are finally processed by a U-Net [37] that generates four output channels, three color channels interpreted as color premultiplied by alpha [35], and a separate alpha channel. We use color values premultiplied by alphas because color in pixels with low alpha tends to be particularly noisy in the extracted mattes and distracts the network when using reconstruction losses on the RGB components.

3.2 Training and Architecture Details

Our network architecture is depicted in Fig. 2. We use the Generative Latent Optimization (GLO) framework [5] to train our network end to end using simple ℓ_1 and perceptual reconstruction losses [17]. We use reconstruction ℓ_1 losses on the premultiplied RGB values, alphas, and a composite on a neutral gray

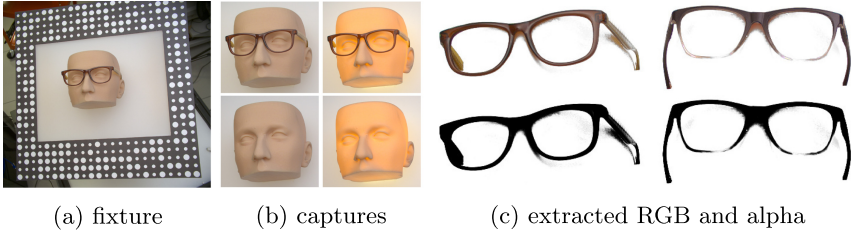| (a) fixture | (b) captures | (c) extracted RGB and alpha |

Fig. 3. (a) Our capture fixture includes a backlit mannequin head and white acrylic plate, surrounded by a Calibu calibration pattern [3], all of which are actuated by a robot arm. We capture (b) four conditions for each pose and object, and solve for (c) foreground alpha mattes and colors. Note some shadows of the eyeglasses remain unmasked, due to limitations of the matting approach. (Color figure online)

background. We also apply a perceptual loss on the composite using the 2nd and 5th layers of VGG pretrained on ImageNet [11]. We found adversarial losses lead to worse results, and we apply no regularization losses on the latent codes.

The latent codes $\mathbf{z}$ for each class are randomly initialized, and we use the Adam [21] optimizer with a learning rate of $1e^{-5}$. We use neural textures of 9 channels, and $\mathbf{z}$ and $\mathbf{w}$ are 8 and 512 dimensions respectively. We generate results at a 512×512 resolution for the eyeglasses dataset and 256×256 for ShapeNet. The latent transformation MLP has 4 layers of 256 features, and the rendering U-Net contains 5 down- and up-sampling blocks with 2 convolutions each, and uses BlurPool layers [47], see more details in the supplementary.

4   Dataset

The *de facto* standard for evaluating category-level object reconstruction approaches is the ShapeNet dataset [7]. Shapenet objects can be rendered under different viewpoints, generating RGB images with ground truth poses, and masks for multiple objects of the same category.

Although using a synthetic dataset can help in analyzing 3D reconstruction algorithms, synthetically rendered images do not capture the complexities of real-world data. To evaluate our approach we acquire a challenging dataset of eyeglasses frames. We choose this object category because eyeglasses are physically small and have well-bounded geometric variations, making them easy to photograph under controlled settings, but they still exhibit complex structures and materials, including transparency, reflections, and thin geometric features.

4.1   Eyeglasses Frames

We collect a dataset of 85 eyeglasses frames under different viewpoints and fixed illumination. To capture the frames, we design a robotic fixture to sample 24×24 viewpoints spanning approximately $\pm 24°$ in yaw and azimuth (Fig. 3a). The fixture includes a Calibu pattern [3] with 3 vertical and 5 horizontal rows, enabling

Table 1. Ablation study comparing multiple baselines on view interpolation of seen instances, and of few-shot reconstruction using $N = 3$ input views, where we fine-tune the whole network together with the latent code. The VAE model is inferior in both tasks, and our approach improves upon DNR in few-shot reconstruction because our textured proxies are not masked by z-buffering.

Model	View interpolation			Few-shot reconstruction		
	VAE	DNR	Ours	VAE	DNR	Ours
PSNR	39.70	41.21	41.32	35.59	36.14	37.19
$PSNR_M$	21.79	23.29	23.42	17.94	18.65	19.64
SSIM	0.9897	0.9916	0.9917	0.9793	0.9819	0.9842
Mask IoU	0.9379	0.9556	0.9556	0.8686	0.8725	0.9012

accurate pose estimation. The fixture center features a hollow 3D printed mannequin head and contains a light inside. For each pose, we capture an image with this backlight on and off (Fig. 3b). We perform difference matting by subtracting the backlit images – which contain fewer shadows – from a reference backlit frame without glasses. We then solve for foreground and background using the closed-form matting approach of Levin et al. [24] (Fig. 3c). The robot's pose is repeatable within 0.5 pixels, enabling precise difference matting.

We generate 3 planar billboards to model each eyeglasses instance: front, left and right. We first compute a coarse visual hull for each object using the extracted alpha masks. We then specify a region of interest in axis-aligned head coordinates, and extract a plane that best matches the surface seen from the corresponding direction. See the supplementary for a more detailed description. We use 5 instances for testing few-shot reconstruction and train on the rest.

Note that this dataset contains two types of artifacts due to the simple acquisition setup: ① shadows cast by the glasses onto the 3D head pollute the alpha mattes and RGB images; ② depending of the viewpoint, the 3D head can occlude part of the glasses frames, resulting in missing temples. We find however that these artifacts do not affect the overall evaluation of our approach.

4.2 ShapeNet

We also train GeLaTO using cars from ShapeNet [7]. We generate the proxies using the auto-encoder version of AtlasNet [15] which takes as input a point cloud. We train a 5 patches/proxies model generating triangular meshes based on a 24×24 uniform grid sampling. Note that the proxies generated by AtlasNet can overlap, but our model is robust thanks to the U-Net compositing step.

5 Evaluation

We evaluate GeLaTO on a number of tasks on the eyeglasses dataset, and then show qualitative results on ShapeNet cars. We compare our representation

Fig. 4. Comparison of view interpolation results for our model and the baselines.
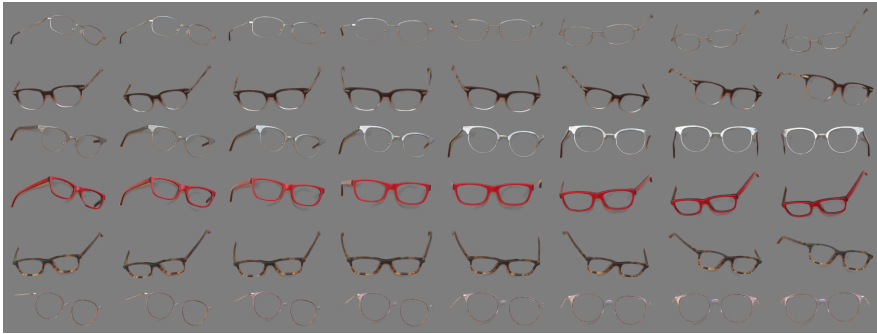

Fig. 5. View interpolation results from our model for a variety of glasses.

against baselines inspired by neural textures [43] using the same proxy geome-
try. In particular, we modify deferred neural rendering (DNR) in two ways: we
parameterize the texture using a generator network, without loss of performance,
and concatenate deep buffer channels consisting of normal and depth informa-
tion to the sampled neural texture, instead of multiplying the sampled neural
texture by the viewing direction vector. A key difference of our method is that
Thies et al. render a deferred rendering buffer with *z-buffering* before the U-Net,
whereas our method *stacks* the deferred rendering buffers of each texture proxy
before the U-Net. Thus our network is able to "see through" transparent layers
to other surfaces behind the frontmost proxy. We evaluate a second baseline
that uses a Variational Auto-Encoder (VAE) [22] instead of GLO [5] to model
the distribution of instances, where the encoder is a MLP that takes as input a
one-hot encoding of the instance id (more details in the supplementary).
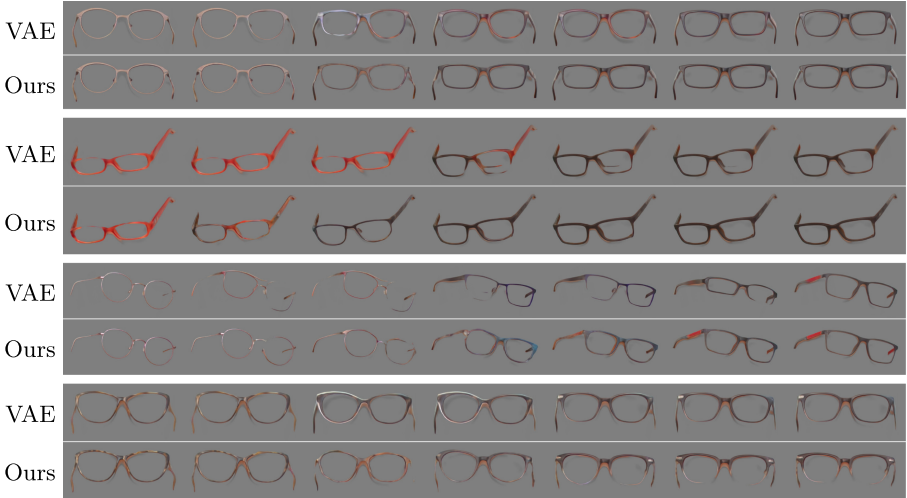
Fig. 6. Examples of instance interpolation of VAE and our model using GLO.

5.1 View Interpolation

We first evaluate our method on the view interpolation task, and show that textured proxies can model complex geometry and view-dependent effects. We train a network on 98% of the views of the training set of the eyeglasses dataset, and test on the remaining 2%. Quantitative results in Table 1 show that our model slightly improves upon the DNR baseline, and is significantly better than VAE. We report PSNR and SSIM on the whole image, $PSNR_M$ evaluated within 7 pixels of alpha > 0.1 values, and IoU of the alpha channel thresholded at 0.5.

Figure 4 qualitatively compares the view interpolation results. VAE results are overly smoothed, and our approach captures more high-frequency details compared to DNR. Figure 5 contains interpolations of the eyeglasses seen from multiple viewpoints, showcasing strong view-dependent effects due to shiny or metallic metallic materials, and reconstructions of transparent glasses that are predominantly composed of specular reflections (last example).

5.2 Instance Interpolation

Our generative model allows interpolations in the latent space of objects, effectively building a deformable model of shape and appearance, reminiscent of 3D morphable models [4]. We visualize such interpolations in Fig. 6, in which the latent code **z** is linearly interpolated while the proxy geometry is kept constant. VAE models are commonly thought to have better interpolation abilities than GLO, because the injected noise regularizes the latent space. However, we find GLO offers better interpolations in our setup. VAE interpolations tend to be less visually monotonic, like in the last example where a white border appears and

then disappears on the left side of the frame, and often contain spurious structures like the double rim on the second example. The supplementary video shows the effects of interpolating the neural texture and proxy geometry independently.
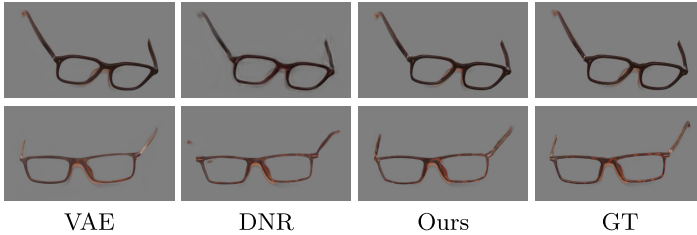

| VAE | DNR | Ours | GT |

Fig. 7. Comparison of few-shot reconstruction using $N = 3$ input views.

Table 2. Reconstruction results with varying numbers of input images N for unseen instances, for the DNR baseline without the category model, finetuning our category-level model, and NeRF. Fine-tuning the category model provides similar quality to DNR with $> 3\times$ fewer input views, and provides $\sim$3 dB improvement with the same number of input views. NeRF generates better results with $N \geq 30$ views, but is significantly slower to train and render novel views.

	DNR [43]		Ours				NeRF [30]			
	Trained from scratch		Finetuning category model				Trained from scratch			
	N = 30	N = 100	N = 3	N = 10	N = 30	N = 100	N = 3	N = 10	N = 30	N = 100
PSNR	38.75	40.05	36.53	39.35	41.61	43.42	31.20	37.21	43.32	**45.28**
PSNR$_M$	21.48	22.43	19.01	21.78	24.00	25.80	15.41	21.25	27.49	**29.80**
SSIM	0.9858	0.9897	0.9824	0.9890	0.9921	0.9942	0.9600	0.9845	0.9947	0.9962
Mask IoU	0.9293	0.9407	0.8864	0.9350	0.9585	0.9682	N/A	N/A	N/A	N/A

5.3   Few-Shot Reconstruction

Because we have parameterized the space of textures, we can think of reconstructing a particular instance by finding the right latent code **z** that reproduces the input views. This can be done either using an encoder network, or by optimization via gradient descent on a reconstruction loss. These approaches are unlikely to yield good results in isolation, because the dimensionality of the object space can be arbitrarily large compared to the dimensionality of the latent space, e.g., when objects exhibit a print of a logo or text. As noted by Abdal et al. [1], optimizing intermediate parameters of the networks instead can yield better results, like the transformed latent space **w**, the neural texture space, or even optimizing all the network parameters, i.e. fine-tuning the whole network.

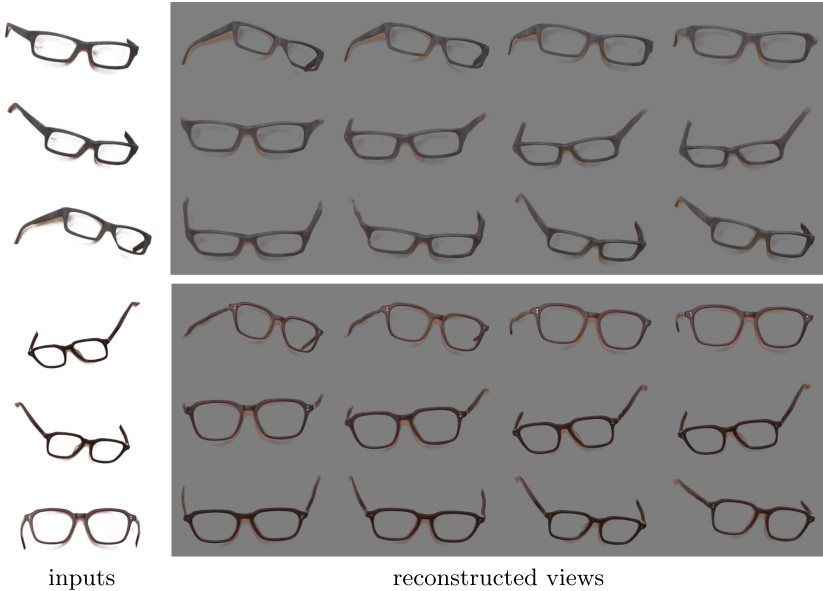inputs   reconstructed views

Fig. 8. Results for few-shot reconstruction using $N = 3$ views. Left: Input views. Right: Reconstructed views using our method after fine-tuning on the input views. Notice that although the first instance is only captured from the left, our network still is able to reconstruct other viewpoints effectively. We are also able to capture view-dependent effects as seen on the bridge region of the glasses.

Thus, given a set of views $\{I_1, \ldots, I_k\}$ with corresponding poses $\{\mathbf{p}_1 \ldots \mathbf{p}_k\}$ and proxy geometry $\{P_1, \ldots, P_K\}$, we define a new latent code $\mathbf{z}$ and set the reconstruction process as optimization

$$\mathbf{z}^\star, \boldsymbol{\theta}^\star = \arg\min_{\mathbf{z}, \boldsymbol{\theta}} \sum_k \|I^k - \text{Net}(\mathbf{z}, \mathbf{p}_k, \boldsymbol{\theta})\|_1,$$

where $\text{Net}(\cdot, \cdot, \cdot)$ is the end to end network depicted in Fig. 2 parameterized by the latent code $\mathbf{z}$, the pose $\mathbf{p}$, and the network parameters $\boldsymbol{\theta}$.

In Table 1, we quantitatively evaluate reconstructions of 5 unseen instances using only $N = 3$ input images, by fine-tuning all network parameters together with the latent code, and show qualitative results in Fig. 7. We use the same baselines as in Sect. 5.1, and report statistics across the 5 instances. We halt the optimization at 1000 steps, because running the optimization to convergence overfits to only the visible data, reducing the performance on unseen views. We observe that the VAE model is inferior, and that stacking the proxy inputs in our model performs better compared to z-buffering in DNR, because the eyeglasses' arms can be occluded by the front proxy, preventing the optimization of the side textured proxy. Figure 8 shows the input images and reconstructed views using our model, illustrating accurate reproduction of view-dependent effects on the bridge and novel views from an unseen side of the glasses.
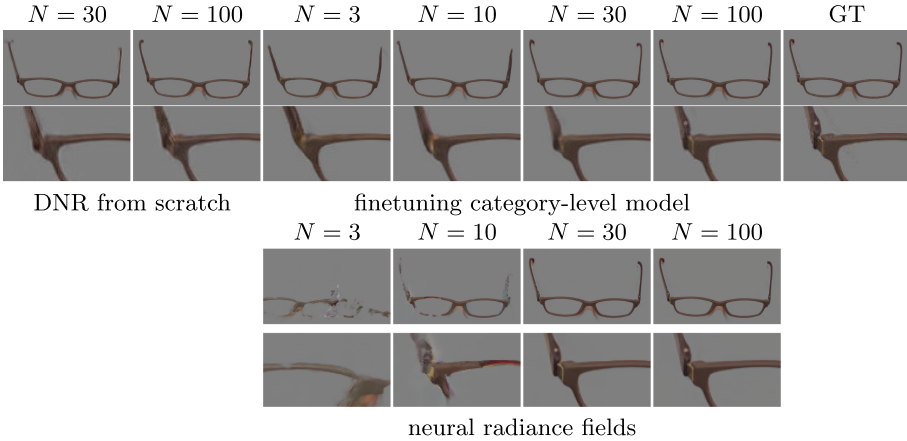
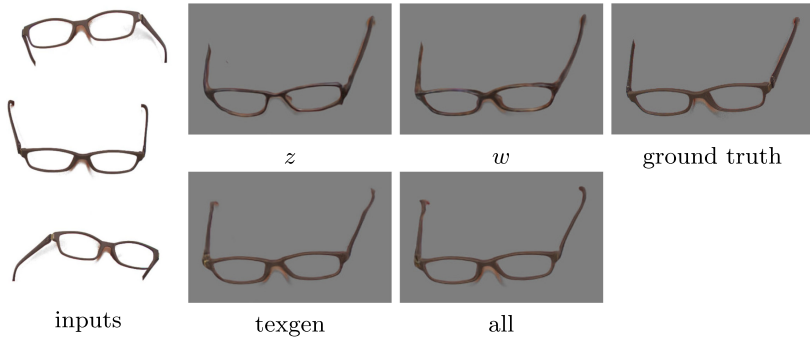Fig. 9. Unseen instance reconstruction varying the number of input images N.

Fig. 10. Differences depending on where the model is being fit. The shape is best fit under **w**, although the texture does not match, and better overall reconstruction is achieved when all network parameters are fine-tuned.

To demonstrate the power of our representation, we compare reconstructions of unseen objects with increasing number of input images N, using our GeLaTO, and the DNR baseline described in Sect. 5.1, that is exclusively trained on the unseen instance. Similar to Thies et al. [43], we optimize the neural texture for 30k and 100k steps for $N = 30$ and $N = 100$ respectively. We also compare with Neural Radiance Fields (NeRF) [30], a concurrent novel-view synthesis technique that uses a volumetric approach that does not require proxy geometry. Table 2 and Fig. 8 show that our representation achieves better results than the DNR baseline with more than 3× less input images. Using the same number of input images, our reconstructions have PSNR score ~3 dB higher than the model trained from scratch. Compared to NeRF, our model is more accurate with few views, although NeRF is significantly better with denser sampling. Moreover, training the DNR baseline takes 50 and 150 min on 15 GPUs for $N = 30$ and
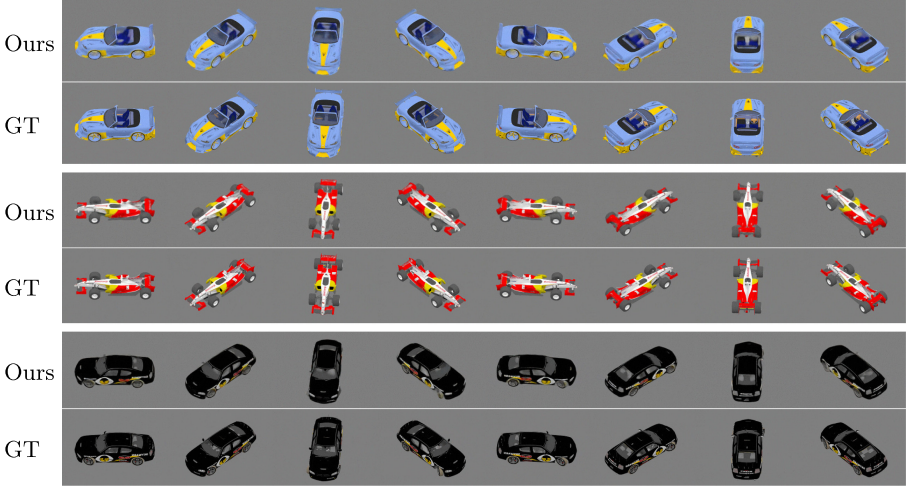
Fig. 11. Reconstruction results on ShapeNet cars using textured proxies based on AtlasNet reconstructions. See supplementary video for more results.

$N = 100$ respectively, whereas fine-tuning GeLaTO takes less than 4 min on a single GPU. Training NeRF takes 4 hours on 4 GPUs and rendering a single using NeRF takes several seconds, making it unsuitable for real-time rendering, while DNR and GeLaTO render new views under 20 ms on a NVidia 1080 Ti.

Table 3. Comparison of reconstructions when fitting in different spaces. z is the instance latent code, w is the transformed latent code, texture refers to fitting also the parameters of the texture generators, and all refers to fine-tuning the neural rendering network as well.

Fit variables	z	w	Texture	All
PSNR	31.30	36.50	37.12	**37.19**
PSNR_M	13.85	18.85	19.59	**19.64**
SSIM	0.9638	0.9833	0.9841	**0.9842**
Mask IoU	0.7242	**0.9152**	0.8984	0.9012

Finally, we evaluate the choice of which variables to optimize during few-shot reconstruction in Table 3, and show comparative qualitative results in Fig. 10. Optimizing the transformed latent code **w** reconstructs the shape best as measured by the mask IoU, albeit with a strong color mismatch. Fine-tuning all the network parameters generates the best results as measured by PSNR.

5.4   Results on ShapeNet

We show results of modeling ShapeNet cars using textured proxies based on AtlasNet reconstructions. We train a model on 100 car instances using 500 views. We use 5 textured proxies, with a 128×128 resolution each, and increase the first layer of the neural renderer from 32 to 64 channels to accommodate the extra proxies' channels. Figure 11 shows unseen view reconstruction results, scoring a PSNR of 30.99 dB on a held-out set.

Fig. 12. Instance interpolations on ShapeNet. Left: reconstructed view of start instance. Middle: latent texture code interpolation while keeping proxy geometry constant. Right: target instance reconstruction using its proxy geometry.
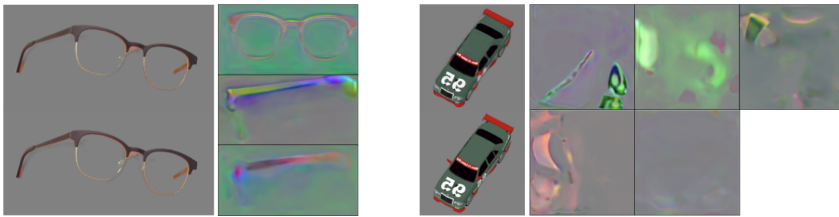

Fig. 13. Learnt neural textures for eyeglasses and cars. Left top: reconstructed view, left bottom: ground truth, right: neural textures. Note the high frequency details encoding the eyeglasses' shape and the number decal on the car.

Figure 12 shows smooth latent interpolation of the latent code of the textured proxies while maintaining the proxy geometry of the first car. Although the proxy geometry is different between instances, Groueix et al. [15] observe that the semantically similar areas of the car are modeled consistently by the same parts
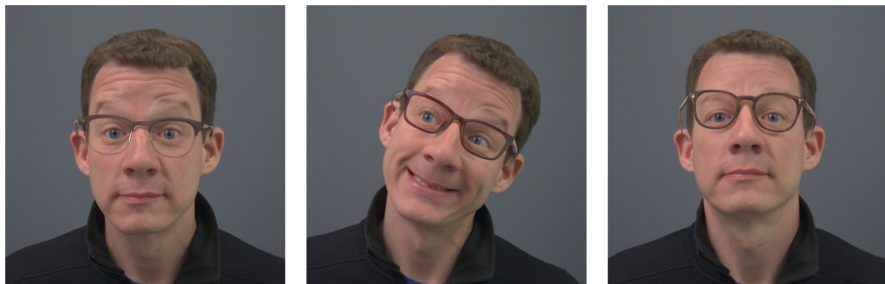
Fig. 14. Virtual try-on application for eyeglasses frames, in which a user without eyewear can virtually place reconstructed glasses on themselves. The eyeglasses are generated by our model given the user's head pose, and composited on the user's view. See supplementary video for more results.

of the AtlasNet patches, allowing our model to generate plausible renderings when modifying only the neural texture. Using the proxy geometry of the first instance creates some artifacts, like the white stripes on the first example that are tilted compared to the car's main axis. The eyeglasses interpolation results are more realistic due to a smaller degree of variability in the object class. Please see the supplementary video for more results.

5.5   Neural Textures

We visualize the learned neural textures in Fig. 13, showing the first three channels as red, green and blue. They contain high frequency details of the object, such as the eyeglasses shape and decals on the car.

5.6   Limitations

Our model has several limitations. When seen from the side, planar proxies almost disappear when rasterized to the target view, creating artifacts on the eyeglasses arms in view interpolations, as seen for a few instances in the supplementary video. Another type of artifacts stems from inaccurate matting in the captured dataset, as seen by the remaining skin color shadows in row 4 of Fig. 4 and the incomplete transparent eyeframe in row 6. In the case of few-shot reconstruction, a major limitation of our model is the requirement of known pose and proxy geometry, which can be tackled as a general 6D pose estimation in the case of planar billboard proxies.

6   Application: Virtual Try-On

Our generative model of eyeglasses frames can enable the experience of virtually trying-on a pair of eyeglasses [46]. Additionally, the learned latent space allows a user to modify the appearance and shape of eyeglasses by modifying the input

latent code. We prototype such a system in Fig. 14, where we capture a video of a user at close distance who is not wearing eyewear, track their head pose using [14], place the textured proxies on the head frame of reference, render the neural proxies to into a RGBA eyeglasses layer and finally composite it onto the frame. Our neural renderer network is sufficiently lightweight – running under 20 ms on a NVidia 1080Ti – that such a system could be made to run interactively.

7   Conclusion

We present a novel compact and efficient representation for jointly modeling shape and appearance. Our approach uses coarse proxy geometry and generative latent textures. We show that by jointly modeling an object collection, we can perform latent interpolations between seen instances, and reconstruct unseen instances at high quality with as few as 3 input images. We show results on a dataset consisting of real images and alpha mattes of eyeglasses frames, containing strong view-dependent effects and semi-transparent materials, and on ShapeNet cars. The current approach assumes known proxy geometry and pose; modeling the distribution of proxy geometry and estimating both its parameters and pose on a given image remains as future work.

References

1. Abdal, R., Qin, Y., Wonka, P.: Image2StyleGAN: how to embed images into the StyleGAN latent space? In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), October 2019. https://doi.org/10.1109/iccv.2019.00453
2. Aliev, K.A., Ulyanov, D., Lempitsky, V.: Neural point-based graphics (2019)
3. Autonomous Robotics and Perception Group: Calibu Camera Calibration Library. http://github.com/arpg/calibu
4. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D faces. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 187–194 (1999)
5. Bojanowski, P., Joulin, A., Lopez-Pas, D., Szlam, A.: Optimizing the latent space of generative networks. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research (2018)
6. Bregler, C., Hertzmann, A., Biermann, H.: Recovering non-rigid 3D shape from image streams. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2000 (Cat. No. PR00662), vol. 2, pp. 690–696. IEEE (2000)
7. Chang, A.X., et al.: ShapeNet: an information-rich 3D model repository. Technical report, arXiv:1512.03012 [cs.GR], Stanford University – Princeton University – Toyota Technological Institute at Chicago (2015)
8. Chen, A., et al.: Deep surface light fields. In: Proceedings of the ACM Computer Graphics Interactive Techniques, vol. 1, no. 1, July 2018
9. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019

10. Décoret, X., Durand, F., Sillion, F.X., Dorsey, J.: Billboard clouds for extreme model simplification. ACM Trans. Graph. **22**(3), 689–696 (2003). https://doi.org/10.1145/882262.882326

11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)

12. Fuhrmann, A., Umlauf, E., Mantler, S.: Extreme model simplification for forest rendering, pp. 57–66, January 2005

13. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE Trans. Pattern Anal. Mach. Intell. **32**(8), 1362–1376 (2010)

14. Google: AR Core Augmented Faces. https://developers.google.com/ar/develop/ios/augmented-faces/overview

15. Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M.: AtlasNet: a papier-Mâché approach to learning 3D surface generation. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

16. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017

17. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43

18. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: ECCV (2018)

19. Kar, A., Tulsiani, S., Carreira, J., Malik, J.: Category-specific object reconstruction from a single image. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015

20. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019. https://doi.org/10.1109/cvpr.2019.00453

21. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

22. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114 (2013)

23. Kulkarni, N., Gupta, A., Tulsiani, S.: Canonical surface mapping via geometric cycle consistency. In: The IEEE International Conference on Computer Vision (ICCV), October 2019

24. Levin, A., Lischinski, D., Weiss, Y.: A closed-form solution to natural image matting. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 228–242 (2008)

25. Liu, L., Chen, N., Ceylan, D., Theobalt, C., Wang, W., Mitra, N.J.: CurveFusion: reconstructing thin structures from RGBD sequences. ACM Trans. Graph. **37**(6), 1–2 (2018)

26. Lombardi, S., Saragih, J., Simon, T., Sheikh, Y.: Deep appearance models for face rendering. ACM Trans. Graph. **37**(4), 1–3 (2018)

27. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: learning dynamic renderable volumes from images. ACM Trans. Graph. **38**(4), (2019). https://doi.org/10.1145/3306346.3323020

28. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: learning 3D reconstruction in function space. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

29. Meshry, M., et al.: Neural rerendering in the wild. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
30. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis (2020)
31. Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: HoloGAN: unsupervised learning of 3D representations from natural images. In: The IEEE International Conference on Computer Vision (ICCV), October 2019
32. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 165–174 (2019)
33. Pittaluga, F., Koppal, S.J., Bing Kang, S., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 145–154 (2019)
34. Pollefeys, M., et al.: Visual modeling with a hand-held camera. Int. J. Comput. Vision **59**(3), 207–232 (2004)
35. Porter, T., Duff, T.: Compositing digital images. SIGGRAPH Comput. Graph. **18**(3), 253–259 (1984)
36. Rohlf, J., Helman, J.: Iris performer: a high performance multiprocessing toolkit for real-time 3D graphics. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994 (1994)
37. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
38. Shan, Q., Agarwal, S., Curless, B.: Refractive height fields from single and multiple images. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 286–293, June 2012
39. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhöfer, M.: DeepVoxels: learning persistent 3D feature embeddings. In: Proceedings Computer Vision and Pattern Recognition (CVPR). IEEE (2019)
40. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: continuous 3D-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems, pp. 1119–1130 (2019)
41. Tewari, A., et al.: State of the art on neural rendering. In: Computer Graphics Forum (EG STAR 2020) (2020)
42. Thies, J., Zollhöfer, M., Theobalt, C., Stamminger, M., Nießner, M.: IGNOR: image-guided neural object rendering. arXiv 2018 (2018)
43. Thies, J., Zollhöfer, M., Nießner, M.: Deferred neural rendering: image synthesis using neural textures. ACM Trans. Graph. **38**(4), 1–2 (2019)
44. Tunwattanapong, B., et al.: Acquiring reflectance and shape from continuous spherical harmonic illumination. ACM Trans. Graph. **32**(4) (2013)
45. Whelan, T., et al.: Reconstructing scenes with mirror and glass surfaces. ACM Trans. Graph. **37**(4) (2018)
46. Zhang, Q., Guo, Y., Laffont, P., Martin, T., Gross, M.: A virtual try-on system for prescription eyeglasses. IEEE Comput. Graph. Appl. **37**(4), 84–93 (2017). https://doi.org/10.1109/MCG.2017.3271458
47. Zhang, R.: Making convolutional networks shift-invariant again. arXiv preprint arXiv:1904.11486 (2019)