In this supplemental material, we show additional texture generation results in Section A, nearest train images by SSIM metric in Section B, texture transfer from query images in Section C, additional comparison with Adobe PhotoShop generative tool D, qualitative visualizations for ablations in Section E, network architecture details in Section F, and additional implementation details in Section G.

A. Additional Generation Results

In Fig. 4 we show additional qualitative evaluation on unconditional texture generation for ShapeNet meshes. Not all baselines are able to produce diverse high-quality textures after training on uncorrelated real-world datasets. As GET3D cannot be successfully trained on sparse views, we use the dense view per-object training provided by the authors; however, this still produces different artifacts. Our learned hybrid mesh-field representation enables Mesh2Tex to generate more realistic textures.

We found that a proxy loss on the coarse (face-based) texture helped with faster convergence and avoiding inconsistent texturing, as our neural field considers only local inputs (Tab. 1). In Fig. 3 we show several qualitative results of generated textures using these ablated models for 'chair' and 'car' categories.

B. Nearest train images by SSIM

Fig. 1 shows the closest train images to the generated textured meshes by SSIM. As train images are real, they do not correspond to the shape geometry of our textured meshes.

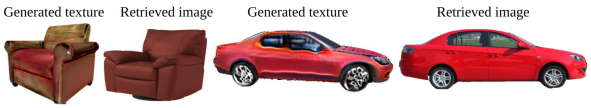

Figure 1: Nearest train images to generated textures with Mesh2Tex by SSIM metric.

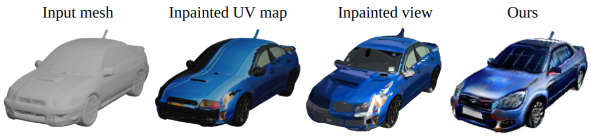

Figure 2: Inpainted texture by PhotoShop generative fill.

Figure 3: Generated textures without global style loss. Supervising only the mesh field ψ generated textures leads to severe mode collapses and inconsistent texture predictions.

Method	Chairs		Cars	
	FID	KID	FID	KID
Ours w/o proxy loss	43.51	2.67	80.87	6.73
Ours	**30.01**	**1.69**	**41.35**	**3.41**

Table 1: Ablation study on unconditional texture generation for 3D shapes. The additional per-face supervision with proxy loss enables more details in texture prediction as our implicit mesh field ψ is able to efficiently refine generated per-face texture. The KID values are scaled by 10^2.

C. Additional Results for Texture Transfer from a Single RGB Image

In Figs. 5, 6, and 7, we present additional results on for texture transfer from synthetic input images, using aligned, unaligned query images and arbitrary shape geometry, respectively. Mesh2Tex effectively leverages our learned texture manifold, preserving consistent textures while transferring to different geometries.

Figs. 8, 9, and 10 present additional results on texture transfer from real-world images from ScanNet and CompCars images as queries. Mesh2Tex is able to perform consistent texture generation from real-world queries even under the challenging scenario of different geometry, pose, and real-world view-dependent effects.

D. Additional Comparison with Adobe PhotoShop Generative Fill

Fig. 2 shows texture inpainting using Adobe PhotoShop [1] generative fill. We project a real RGB image query to the left side of the input car mesh, and inpaint either the UV map or the 2D view of the object; both result in view and geometry inconsistencies.

E. Qualitative Ablation Visualizations

According to Tab.8 in the main paper, we show qualitative results on an ablation study performed in texture generation from unaligned synthetic ShapeNet images. Optimizing textures without patch loss component or NOC guidance leads to messy textures with stripe artifacts and disordered texture mapping. Optimizing only the latent codes (w/o surface features) results in inaccurate texture generation with lost details.

F. Network Architecture

An overview of the architectures of our surface encoder, generator, and neural field Ψ is shown in Figs. 12 and 13. Note that we follow the use of FaceResNet blocks, FaceConv layers, and Synthesis Block from Texturify [5]. Our generator then produces both coarse per-face rgb values as well as feature vector F_c input to Ψ. Our discriminator architecture

follows the discriminator of Texturify.

In order to produce locally refined texture with Ψ, we operate locally on faces, considering their barycentric coordinate system. We compute per-vertex features from F_c by averaging incident face features. For a point p on the mesh surface, its feature is computed as the barycentric averaging of the vertex features F_1, F_2, F_3, where the barycentric weights b_1, b_2, b_3 are areas of triangles $\triangle\ (F_1, F_2, p), \triangle\ (F_2, F_3, p), \triangle\ (F_3, F_1, p)$ respectively. Ψ also takes a learnable auxiliary latent vector of size $z_{aux} = 512$ as input, which is fixed for the entire model. We observed that this auxiliary latent enhanced the consistency of high-resolution textures. The auxiliary latent, along with surface encoder and generator features, are then processed with two linear layers (LeakyReLU activations) before being concatenated and passed to an additional four linear layers. This results in the final output color corresponding to surface location p.

When optimizing for texture from an input image query, we also use a pose predictor network and NOC predictor network. Pose prediction uses a ResNet18 [3] backbone, with the final features of size 512 passed to two linear layers with output dimension 256. This refined feature is then passed to a two-layer MLP with hidden size 128 to estimate the angles α^a and α^e. The NOC predictor leverages the EfficientNet-b4 [7] architecture as a backbone for the U-shaped model. It takes an RGB image and the corresponding binary mask of a foreground object as 4-channel input and predicts NOCs a 3-channel image.

G. Implementation Details

We train Mesh2Tex using an Adam optimizer with learning rates of 1e-4, 12e-4, 1e-4, 14e-4 for the encoder, generator, Ψ, and both discriminators, respectively, for PhotoShape [4], and learning rates of 1e-4, 15e-4, 5e-4, 1e-4 for CompCars [8]. For both models, we use a batch size of 2, and render 8 views for each shape in the batch.

To optimize texturing for input query images, we optimize latent codes for 100 iterations and refined weights with the parameters of the two last generator synthesis blocks for additional 300 iterations using an Adam optimizer with a learning rate of 1e-2. For the style loss, we use VGG19 [6] network, and extract features of the $(2^{nd}, 4^{th}, 8^{th}, 12^{th}, 16^{th})$ convolutional layers. We extract patches of size 64 from both rendered and input images. Both full images and patches are equipped with corresponding foreground masks to filter out extracted VGG background features.

Pose prediction is trained using an Adam optimizer with a learning rate of 3e-4 for chairs and 2e-5 for cars, with a batch size of 128 on images of size 512x512. The NOC Predictor is trained using an Adam optimizer with a learning rate of 3e-4 for chairs and 5e-5 for cars, with a batch size of 64 on images of size 512x512. Both networks are first pretrained on synthetic renders of ShapeNet objects and fine-tuned on real-world ScanNet and CompCars datasets (except for NOC Prediction for CompCars as NOC data is not available).

References

[1] Adobe Inc. Adobe photoshop. 1

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[4] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 37(6), Nov. 2018. 2

[5] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces, 2022. 1

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2

[7] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 2

[8] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3973–3981, 2015. 2

Figure 4: Additional results on unconditional texturing for meshes from ShapeNet [2], in comparison with state of the art. Our approach generates more realistic, detailed textures.

Figure 5: Optimized textures based on input query images (top row) using aligned query images from ShapeNet chairs and cars.

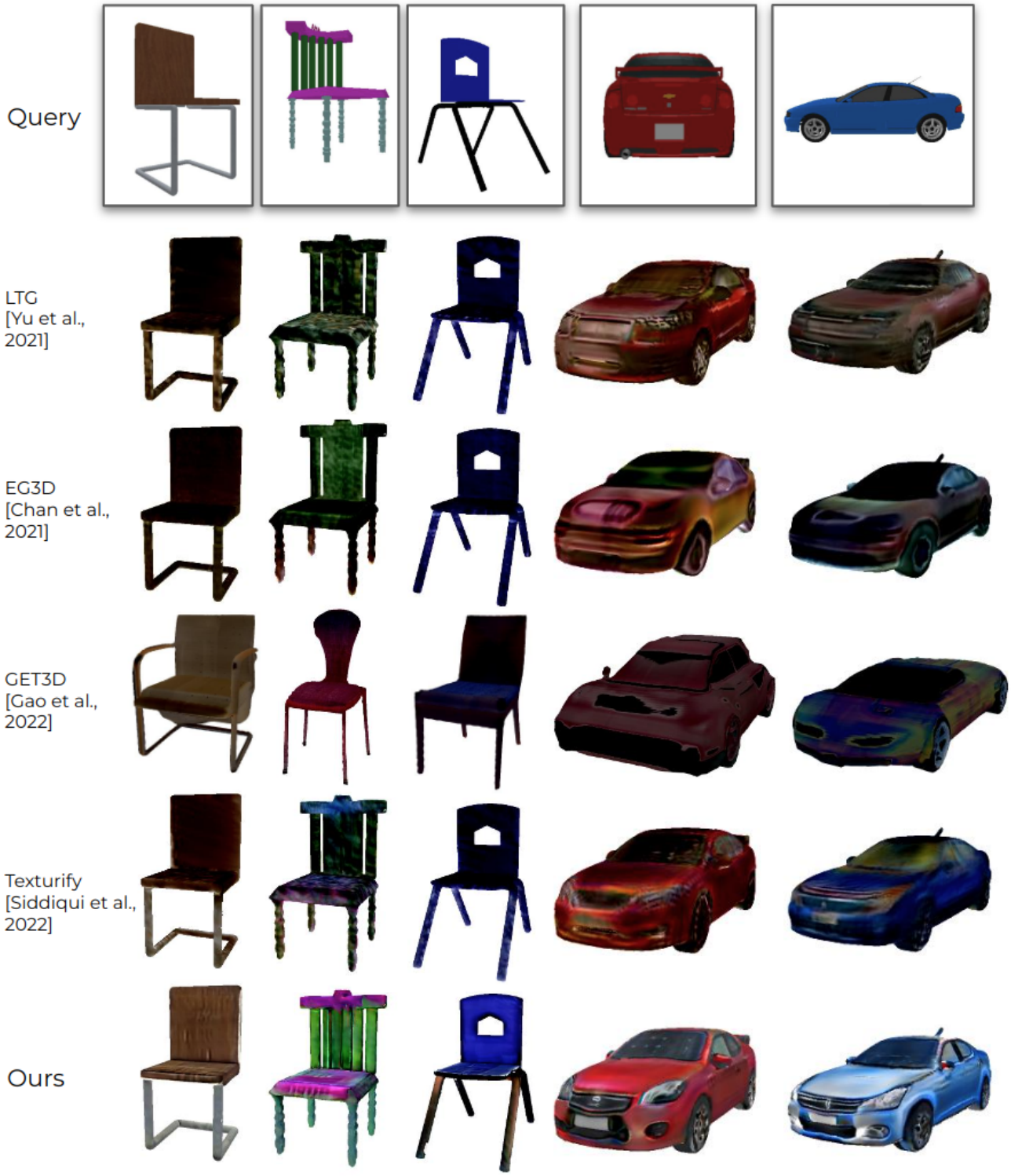

Figure 6: Optimized textures based on input query images (top row) using unaligned images query images from ShapeNet chairs and cars.

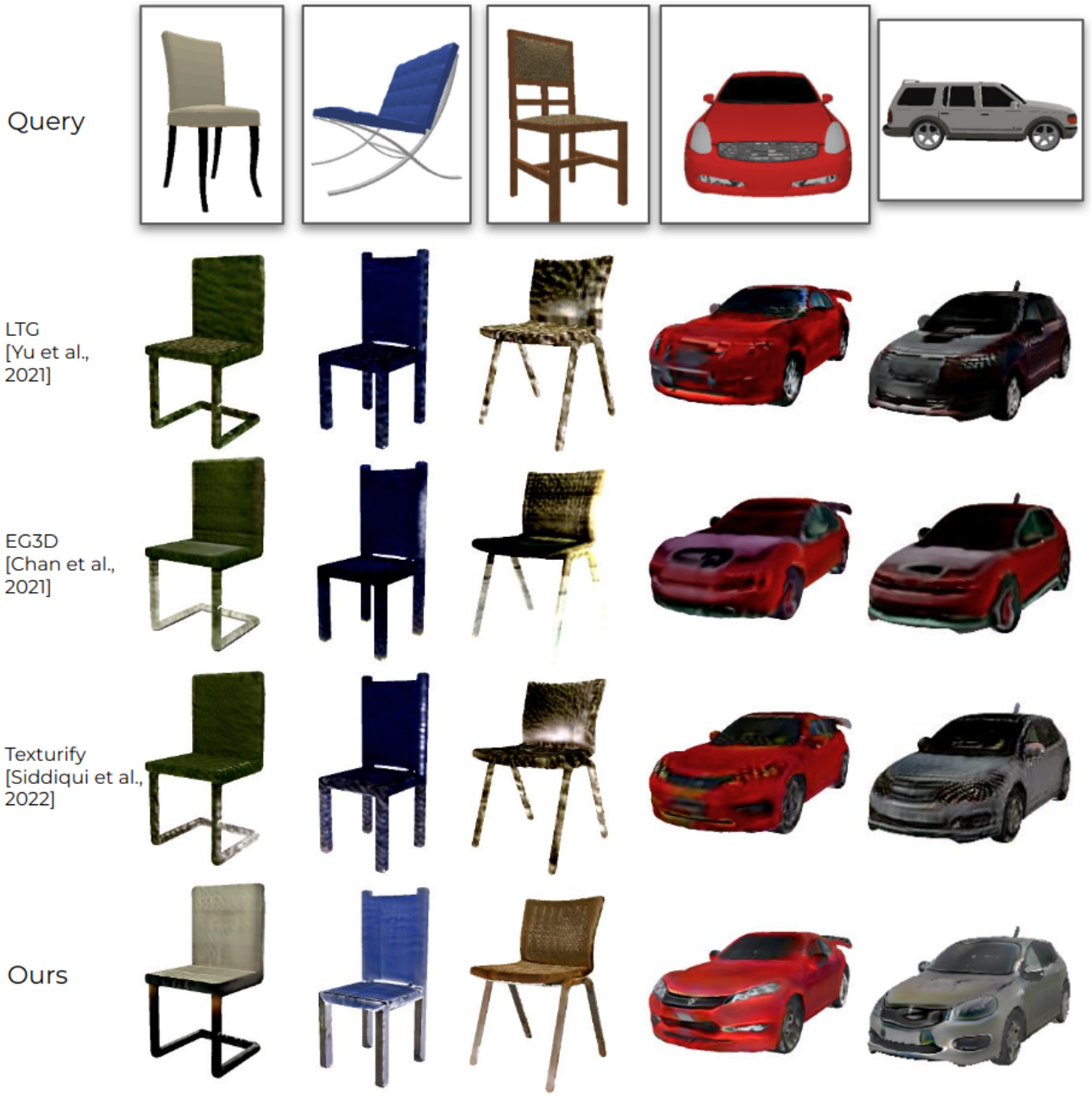

Figure 7: Texture transfer from input query images (top row) using unaligned images and arbitrary shape geometry of the same class category (ShapeNet).

Figure 8: Texture transfer from real-world input query images (top row, ScanNet) using aligned images and close shape geometry (ShapeNet).

Figure 9: Texture transfer from real-world input query images (top row, ScanNet) using unaligned images and similar shape geometry (ShapeNet).

Figure 10: Texture transfer from real-world input query images (top row, ScanNet, CompCars) using unaligned images and arbitrary shape geometry from the same class category (ShapeNet).

Figure 11: Qualitative ablation study on texture transfer from synthetic input query images (ShapeNet) using unaligned images; visualized with the query image pose.

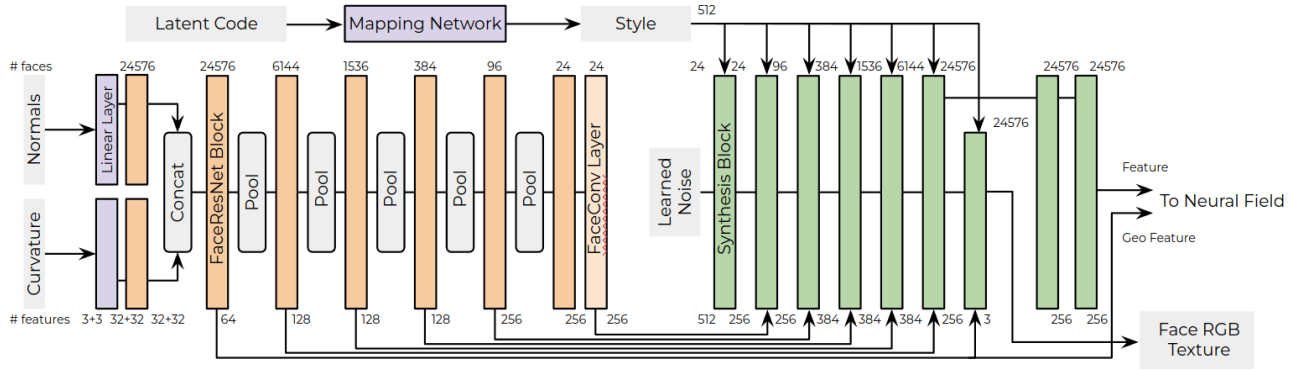

Figure 12: Overview of Surface Encoder (orange) and Generator (green) architectures. The encoder takes as input normals and curvatures of the finest resolution quadmesh, processes them in a hierarchical convolutional structure to extract geometric features. The generator then considers a latent texture code, learned noise, and the geometric features to produce per-face features for the neural field.

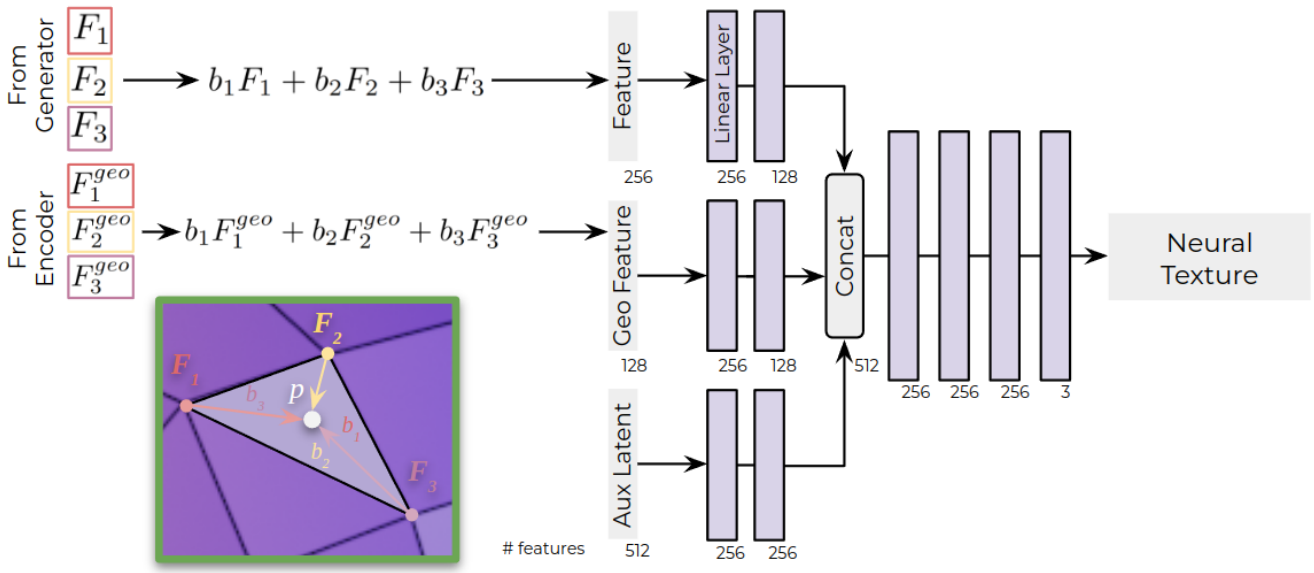

Figure 13: Our architecture for local face neural field Ψ. Features from the surface encoder and generator are fused by their barycentric coordinates and passed with an auxiliary latent vector into an MLP to produce the final color of surface location p.