Mesh2Tex: Generating Mesh Textures from Image Queries

Alexey Bokhovkin
Technical University of Munich
aleksei.bokhovkin@tum.de

Shubham Tulsiani
Carnegie Mellon University
shubhtuls@cmu.edu

Angela Dai
Technical University of Munich
angela.dai@tum.de

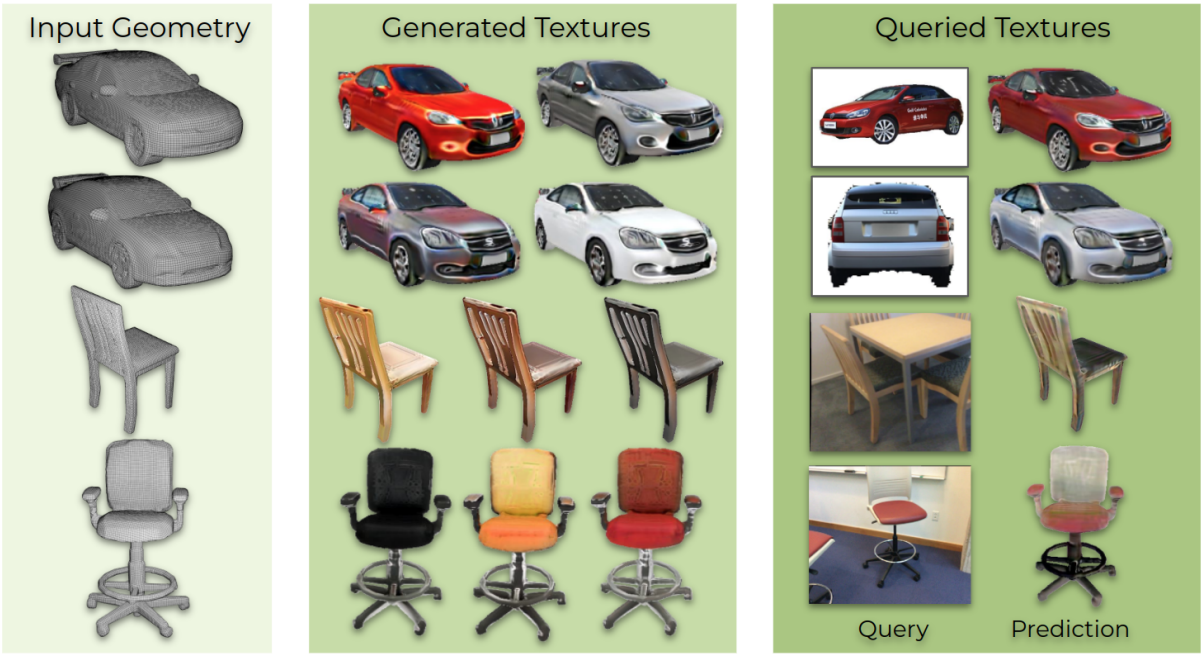

Figure 1: Mesh2Tex learns realistic object texturing on a shape geometry through a hybrid mesh-field texture representation supporting high-resolution texture generation on various shape meshes. Furthermore, our learned texture manifold supports texture transfer optimization from image queries without requiring any matching geometry or pose alignment to the image, producing perceptually consistent texturing in this challenging content creation scenario.

Abstract

Remarkable advances have been achieved recently in learning neural representations that characterize object geometry, while generating textured objects suitable for downstream applications and 3D rendering remains at an early stage. In particular, reconstructing textured geometry from images of real objects is a significant challenge – reconstructed geometry is often inexact, making realistic texturing a significant challenge. We present Mesh2Tex, which learns a realistic object texture manifold from uncorrelated collections of 3D object geometry and photorealistic RGB images, by leveraging a hybrid mesh-neural-field texture representation. Our texture representation enables compact encoding of high-resolution textures as a neural field in the barycentric coordinate system of the mesh faces. The learned texture manifold enables effective navigation to generate an object texture for a given 3D object geometry that matches to an input RGB image, which maintains robustness even under challenging real-world scenarios where the mesh geometry approximates an inexact match to the underlying geometry in the RGB image. Mesh2Tex can effectively generate realistic object textures for an object mesh to match real images observations towards digitization of real environments, significantly improving over previous state of the art.

Project page: alexeybokhovkin.github.io/mesh2tex/

1. Introduction

The ability to obtain 3D representations of real-world objects lies at the core of many important applications in graphics, robotics, movies and video games, and mixed reality. Approaches tackling the tasks of 3D generation [31, 24, 19, 27, 28, 9] or single-view reconstruction [5, 25, 45, 23, 6] enable users to easily create digital 3D assets, either from scratch or conditioned on an image. However, these approaches primarily focus on generating/inferring the geometry of objects, and this alone is insufficient to capture realistic environments, which requires high-quality texturing.

In this work, we propose Mesh2Tex tackle the complementary task of high-fidelity texture generation given known object geometry. In addition to allowing texturing generation for a given object mesh, Mesh2Tex also enables image-based texture synthesis – synthesizing textures that perceptually match a single RGB image observation. In contrast to existing appearance modeling works that characterize texture as a field over the volume of 3D space [3, 37, 38, 29], we observe that object textures lie on object surfaces, and instead generate textures only on the mesh surface, with a hybrid explicit-implicit texture representation that ties the texture field to the barycentric coordinates of the mesh faces. This produces textured meshes directly compatible with downstream applications and 3D rendering engines.

As large quantities of high-quality textured 3D objects are very expensive to obtain, requiring countless hours of skilled artists' work, we train our texture generator with uncorrelated collections of 3D object geometry and photorealistic 2D images, leveraging differentiable rendering to optimize mesh textures at arbitrary resolutions to match the quality of real 2D images in an adversarial fashion. That is, we generate coarse features and colors for each mesh face, which are then refined to high-resolution textures through a neural field defined on the barycentric coordinates of each mesh face. This hybrid texture representation enables differentiable rendering for texturing on explicit mesh surfaces while exploiting the efficiency of neural field representations.

Our learned texture generator can then be used to produce realistic textures to match an input RGB observation of an object, by optimizing over our learned texture manifold conditioned on the mesh geometry, to find a latent texturing that perceptually matches the RGB image. As state-of-the-art object geometry reconstructions are typically inexact (e.g., noise, oversmoothing, retrieval from a database), our learned texture manifold enables effective regularization over plausible textures which effectively capture the perceptual input while providing realistic texturing over the full object. We formulate a patch-based style loss to capture perceptual similarities between our optimized texture and the RGB image, guided by dense correspondence prediction to correlate them. Experiments demonstrate that we outperform state of the art in both unconditional texture generation as well as image-conditioned texture generation.

In summary, our contributions are:

- a new hybrid mesh-neural-field texture representation that enables diverse, realistic texture generation on object mesh geometry by tying a neural texture field to the barycentric coordinate system of the mesh faces. This enables learning a rich texture manifold from collections of uncorrelated real images and object meshes through adversarial differentiable rendering.
- our learned texture manifold enables effective inference-time optimization to texture an object mesh to perceptually match a single real-world RGB image; crucially, we maintain robustness to real-world scenarios of differing views and even geometry with a patch-based perceptual optimization guided by dense correspondences.

2. Related Work

Optimization-based Texturing. Various approaches have been proposed to texture 3D shapes by optimizing from aligned image views of the shape. The traditional optimization approach of Zhou et. al. [49] solves a global optimization for a mapping from observed RGB images onto reconstructed geometry. More recently, Huang et. al. [15] introduce a modern adversarial discriminator loss, coupled with differentiable rendering, to optimize for texture on reconstructed geometry to produce realistic texturing robust to camera pose misalignments in real-world RGB-D capture scenarios. While these methods can produce impressive texturing results, they require aligned 2D/3D data and cannot produce textures in unobserved regions.

Learned Texture Completion and Generation. Learning-based methods have also explored generative texturing, aiming to learn generalized features for unconditional synthesis or conditional inpainting. Photoshape [32] proposes to learn material retrieval to apply materials to 3D shapes. Recently, several methods have been developed to generate textures on various representations: SPSG [8] produces per-voxel colors with an adversarial differentiable rendering formulation, Pavollo et. al. [33] learn texturing in the UV domain, and Henderson et. al. [13] and Texturify [40] propose to generate per-face mesh colors with variational and adversarial training, respectively. The recent success of neural implicit field representations has also inspired implicit texture field models [30, 3, 10]. Our approach aims to model texture on the surface of a mesh, coupled with the representation capacity of a shared implicit field per mesh face.

Query-based 3D Understanding. The recent success of vision-language models such as CLIP [34] has sparked interest in query-guided 3D understanding from text and images.

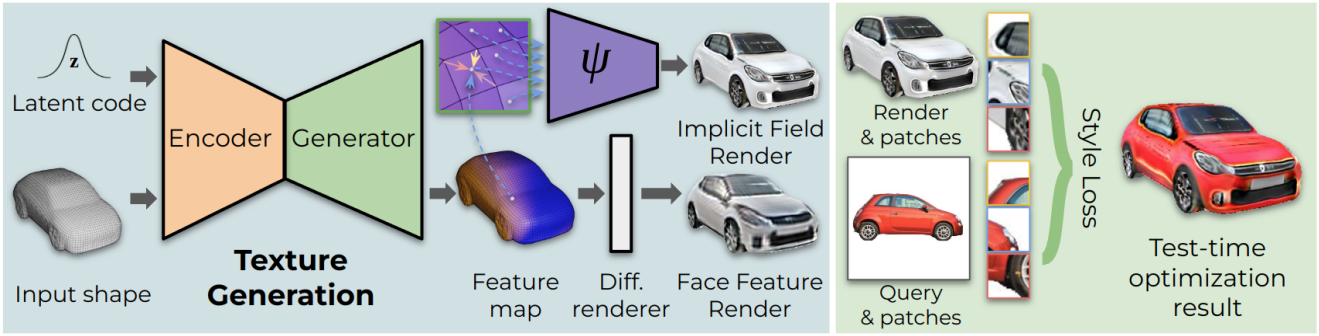

Figure 2: Method overview. Left: we first learn a texture generator for textures represented as a shared local neural field in the barycentric coordinate system of each mesh face. This enables the capture of high-resolution texture detail, while remaining tied to surface geometry, avoiding ambiguities of volumetric field representations. Our learned texture manifold can then be used for texture transfer from a single RGB query image, based on a correspondence-guided patch-based style loss to produce perceptually consistent texturing even for shapes with differing geometry and unknown image pose.

Text and image guidance have been exploited for knowledge distillation for 3D semantic understanding [22, 36, 18]. Various shape generation tasks have also been formulated with text queries, by leveraging CLIP as supervision [39, 17, 43]. In particular, Text2Mesh [26] leverages CLIP guidance for text-based texture optimization on 3D shapes. We formulate Mesh2Tex to support image-based queries for texture transfer optimization to various 3D shapes, to reflect potential practical texturing scenarios where images are largely easy to capture or find, with high visual specificity. We thus tackle challenges in geometric and pose misalignments from various potential image queries.

3. Method

3.1. Overview

We aim to learn a high-quality texture manifold conditioned on a given shape mesh geometry, which supports unconditional texture generation as well as optimization within the texture manifold to fit to image queries for practical texturing scenarios. An overview of our approach is visualized in Figure 2. We propose to learn this texture manifold on a hybrid representation that combines a mesh representation with a neural field operating on the barycentric coordinates of each mesh face. Our texture generator then generates initial coarse features per-face for a given shape mesh, which are then refined by a shared neural field across the mesh faces. This enables high-resolution texture generation by sampling the neural field on the mesh surface. In the absence of a large-scale dataset of high-quality 3D textured shapes, we supervise the texture generation with an adversarial loss with photorealistic 2D images, through differentiable rendering.

We can then traverse our learned texture manifold to generate textures for a given shape geometry to perceptually match to image queries. Since an RGB image query may be taken at an arbitrary camera pose, we estimate the object pose in the RGB image along with its normalized object coordinates [44] to guide the texture optimization. Furthermore, we must also support texture transfer scenarios, as having an exact geometric shape match for an arbitrary RGB image query cannot be assumed in practice. We thus employ both a global and local patch-based style loss to generate a realistic texture that perceptually matches the query image.

3.2. Mesh-Field Texture Generation

We learn a texture manifold on a mesh-field representation. Given a shape mesh $\mathcal{M}$, we use a latent-variable conditioned encoder-decoder to obtain per-face features F. We then employ a shared neural field ψ that operates on each face to produce arbitrary-resolution texturing. High-quality texture generation is then learned by supervision with photorealistic 2D images in an adversarial fashion through differentiable rendering. Our texture generator training is shown in Figure 3.

A shape mesh $\mathcal{M}$ is represented as a quadrilateral mesh; inspired by the hierarchical mesh representation of Texturify [40], we also employ QuadriFlow [16] to parameterize shape meshes as hierarchies of 4-way rotationally symmetric (4-RoSy) fields. We can then leverage convolutional and pooling operators on the neighborhoods of each quad face, enabling encoding and decoding features on such mesh faces with arbitrary topologies. We employ an encoder-decoder face convolutional network G on $\mathcal{M}$, which takes as input geometric quad face features (per-face normals, fundamental forms, and curvature), and predicts per-face features F_c. Similar to StyleGAN [20] we incorporate a latent texture code z into the decoder through a mapping network.

As we aim to generate high-quality textures not limited to the mesh face resolution, we leverage a neural field ψ. ψ takes as input a barycentric coordinate p on a mesh face, along with mean face feature F_v of all incident quad faces:

$$\psi(p, b_1 F_{v_1} + b_2 F_{v_2} + b_3 F_{v_3}) = c \in \mathbf{R}^3_{[-1,1]}, \quad (1)$$

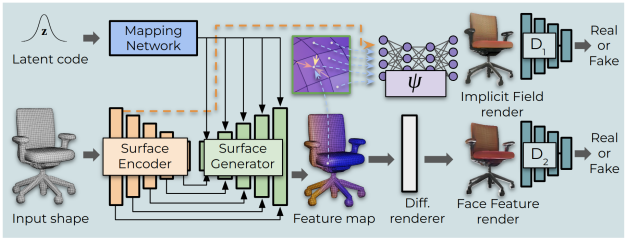

Figure 3: Mesh-field texture generation. We employ face convolutions following [40] for initial coarse feature generation, followed by a neural field that interprets the per-face features to produce refined color outputs at arbitrary locations on the mesh surface. The hybrid mesh-field texture generation is supervised in an adversarial fashion through differentiable rendering, with photorealistic 2D images representing the real distribution.

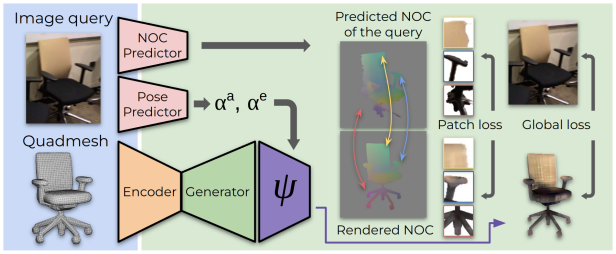

Figure 4: Texture transfer optimization from a single image. We traverse our learned texture manifold for texture transfer from a single RGB image query. Since image alignment may be unknown, we estimate a coarse pose along with finer-grained dense NOC correspondences. We leverage the NOCs to guide patch sampling for our patch-based style loss to produce texturing that perceptually matches to the query images, without requiring any exact geometric shape matches to the image.

where $\mathcal{A} = \{v_1, v_2, v_3\}$ is the triangle half of a quad face $\{v_1, v_2, v_4, v_3\}$ in which p lies, $\{b_1, b_2, b_3\}$ are the barycentric coordinates of p with respect to triangle $\mathcal{A}$, and c is the final output color. We can then generate arbitrary-resolution textures by densely sampling ψ on each mesh face.

To supervise the learning of this mesh-field texture manifold, we differentiably render our generated textures for an adversarial loss with photorealistic 2D images. To render the generated textures, we sample ψ at the locations corresponding to each pixel in the rendered view. In addition to implicit field ψ rendering, we also pass the output feature map F_c into one convolutional layer and render it using PyTorch3D [35] differentiable renderer and supervise it similarly as a proxy loss. We then train the texture manifold in an end-to-end fashion using a non-saturating GAN loss with gradient penalty and path length regularization.

3.3. Texture Transfer from a Single Image

Once learned, we can not only use our texture manifold for unconditional generation, but also importantly traverse through the manifold to produce shape texturing that matches to a query RGB image I. This represents real-world texturing scenarios where a user may wish to texture a shape based on an easy-to-capture image inspiration. Here, notable challenges are lack of knowledge of the ground truth object pose in I, and handling inexact geometric matches between the shape mesh to be textured and the object in I – as it is not practical, nor always desirable, to assume the ability to reconstruct or retrieve the exact geometry of the object in an arbitrary image. We thus aim to produce textures on a shape to perceptually match an image I in these challenging scenarios. Our texture transfer optimization is illustrated in Figure 4.

From I, we first estimate the object pose as the azimuth and elevation angles, α^a and α^e, respectively. To this end, we use a ResNet-18 [12] network on I, pre-trained on synthetic rendered shapes to classify α^a and α^e into 12 and 5 bins for a coarse rotation estimate. For finer-grained reasoning, we predict normalized object coordinates [44] I_{NOC}, dense correspondences from the object pixels in I to the canonical space of the object. For NOC prediction, we use a UNet-style network with an EfficientNet-b4 [42] backbone, trained on synthetic rendered shape data.

We then formulate our texture transfer optimization. We differentiably render our generated texture from initial pose estimate α^a and α^e to produce generated image X. Since we optimize for perceptual similarity, we employ a style loss [11] in a global and local fashion to compare X and I.

We then optimize for latent texture code z with the loss:

$$L = w_{glob} \sum_{h=1}^{N_h} \sum_{l=1}^{N_l} L_{glob}(I^{h,l}, X^{h,l}) + w_{patch} \sum_{p=1}^{N_p} \sum_{h=1}^{N_h} \sum_{l=1}^{N_l} L_{patch}(I_p^{h,l}(x,y), X_p^{h,l}(x',y')), \tag{2}$$

where L_{glob} denotes a global style loss, L_{patch} a local patch-based style loss, and w_{glob} and w_{patch} constants to balance the loss values.

Global style loss. L_{glob} considers the full images I and X at $N_h = 3$ resolution levels (original, half, and quarter resolution): I^h and $X^h, h \in [1, 2, 3]$. The global style loss is then computed on each pair (X^h, I^h) on $N_l = 5$ VGG feature layers.

Patch style losses. Since the precise structure of the object in I may not match with that of X (e.g. geometry mismatch or inconsistent pose prediction), we further employ local

style losses on image patches. We extract $N_p = 2$ patches $I_p(x, y)$ from I randomly per iteration, located at patch centers (x, y) within the object. We then use the corresponding NOC estimate at the patch location, $I_{\text{NOC}}(x, y)$, and find a corresponding patch $X_p(x', y')$ where (x', y') is the pixel location whose shape NOC is closest to $I_{\text{NOC}}(x, y)$ w.r.t. ℓ_2 distance. Note that since X has been rendered from a shape mesh, we can use ground truth shape NOCs for X, and only require predicted NOCs for I. This provides a coarse correspondence between patches to guide the style loss. Similar to L_{glob}, we apply L_{patch} at $N_h = 3$ resolution levels and with $N_l = 5$ VGG feature layers.

Face feature refinement. With texture code z optimized from Equation 2, we further allow for texture refinement by unfreezing the last two face convolutional layers in the texture generator. We optimize for refined weights in the surface features using only L_{patch}, which enables better capturing of local detail in I.

Implementation Details Our Mesh2Tex model is implemented using PyTorch and trained using an Adam [21] optimizer with learning rates of 1e-4, 12e-4, 1e-4, 14e-4 for the encoder, generator, neural field ψ parameters and both discriminators, respectively. Mesh2Tex is trained on 2 NVIDIA A6000s for 50k iterations ($\sim$ 100 hours) until convergence.

At test-time, we optimize latent codes for 100 iterations and refined weights for additional 300 iterations, which takes $\sim$ 400s in total. During texture optimization, we extract 64 $\times$ 64 patches. We provide further details in the supplemental.

4. Results

We evaluate the texture generation capabilities of Mesh2Tex on unconditional generation and image-based texture generation, on both synthetic and real data. For both scenarios, our texture generation is trained from real images.

Datasets. For evaluation, we use object geometry from ShapeNet [4] for texturing, and real-world query images from ScanNet [7] for chairs, and from CompCars [46] for cars. For synthetic experiments requiring exact geometric matches, we use ShapeNet textured objects and render query image views. For real image queries with close matching geometry, we use Scan2CAD [1] annotations for chair meshes to ScanNet images. For CompCars, we use the coarse pose information (front, back, left, right, etc.) to estimate close image alignments.

Note that all methods, except GET3D, were trained with the same set of ShapeNet meshes, and images from PhotoShape [32] and CompCars [46] for chairs and cars. GET3D requires a much denser sampling of images per shape rather than a single view per shape, so we instead compare with the authors' released pre-trained GET3D model.

Figure 5: Unconditional texturing for meshes from ShapeNet [4], in comparison with state of the art. Our approach generates more realistic, detailed textures.

Evaluation metrics. We evaluate the perceptual quality of our generated textures with several metrics. To measure the realistic quality of the textures, we compare rendered views of generated textures on various shapes to real-world images, using the standard FID [14] and KID [2] scores used for assessing generated image quality. For FID and KID evaluation, we use real images from PhotoShape [32] and CompCars [46] for chairs and cars, respectively, and 6 rendered views from each synthesized textured shape. Additionally, for texture transfer from a single image, we compute CLIP [34] similarity as cosine distances between the query image and rendered texture. In synthetic experiment setups where exact geometric matches are available, we further compute an LPIPS [48] perceptual metric between synthesized views and the query image.

Baselines. We compare with several state-of-the-art texturing methods leveraging various texture representations: Yu et. al. [47] learns texture generation on a UV map parameterization, EG3D [3] leverages an efficient triplane representation for view synthesis, Texturify [40] generates per-face colors on a mesh, and GET3D [10] jointly estimates color and geometry with triplane representations.

4.1. Unconditional Texture Generation

Table 1 and Figure 5 show quantitative and qualitative comparisons of unconditional texture generation to state of

Method	Parameterization	Chairs		Cars	
		FID	KID	FID	KID
EG3D	Tri-plane Implicit	36.45	2.15	83.11	5.95
Yu et al.	UV	38.98	2.46	73.63	5.77
GET3D	Tri-plane Implicit	46.20	3.02	89.62	6.92
Texturify	4-RoSy Field	31.08	1.75	59.55	4.97
Ours	4-RoSy Implicit Field	**30.01**	**1.69**	**41.35**	**3.41**

Table 1: Unconditional texture generation for 3D shapes. Our mesh-field approach outperforms state of the art in generation quality. The KID values are scaled by 10^2.

the art. Our hybrid mesh-field texture representation enables richer texture generation with finer-scale details than state-of-the-art baselines.

4.2. Texture Transfer from a Single Image

We further evaluate texturing of objects from image queries, which opens up new possibilities in the content creation process. We consider both synthetic image queries, which enable texturing of exact geometric matches, as well as real image queries, where no object geometry matches exactly to the queries. For all baselines, we apply a similar texture optimization as our approach, leveraging global and patch-style losses without NOC guidance.

4.2.1 Synthetic Image Query Experiments

Exact geometry and known image alignment. We evaluate texture generation from a single image query to exactly matching object geometry with known image alignment in Tables 2 and 3, which measure the distribution quality and perceptual match to the query image, respectively. The optimization through our high-fidelity texture manifold enables more detailed texturing to match to various texture changes in the query image, in comparison to baselines.

Exact geometry and unknown image alignment. We evaluate texture generation from a single image query where the pose of the shape in the input query is unknown. Quantitative evaluation with state of the art is shown in Tables 4 and 5, which measure the distribution quality and perceptual match to the query image, respectively. Figure 6 visualizes qualitative comparisons. For GET3D, we do not evaluate LPIPS, as it can produce geometric changes resulting in an inexact geometry match to the query.

Even with exact geometry, the unknown image alignment poses a notable challenge, resulting in degraded performance in the baselines. Our approach maintains more robustness due to the NOC guidance in our patch loss during texture optimization, producing more plausible texture generation.

Texture transfer. We evaluate texture transfer experiments from a single image query to an arbitrary shape of a different geometry than the input query. Tables 6 and 7 compare our

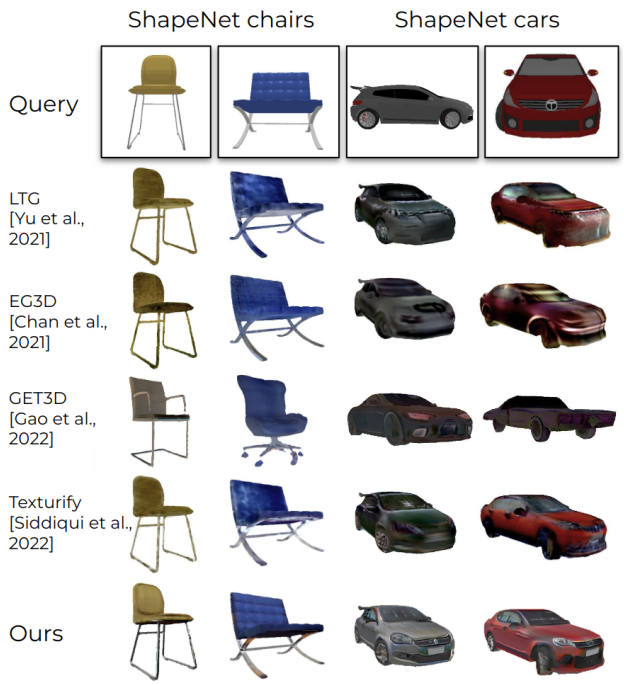

Figure 6: Texture transfer from synthetic image queries to exact shape geometry with unknown image alignment. Mesh2Tex produces more geometrically consistent texturing with finer-resolution details captured in our mesh-field texture representation. Note that GET3D models both geometry and color, resulting in possible geometric changes.

approach to state of the art. This challenging scenario of differing geometry results in a performance drop in all methods; however, due to our NOC-guided style optimization on a mesh-field texture representation, Mesh2Tex can produce textures on novel shape geometry with greater coherency and finer details than baselines.

4.2.2 Real Image Query Experiments

Closely-matching geometry and image alignment. We evaluate texturing from a single real-world image, where the shape geometry to be textured and image alignment are close (as we cannot guarantee an exact match to real-world images). We compare with state of the art in Tables 2 and 3. In the real-world setting, view-dependent effects in real images (e.g., specular highlights, reflections) pose a notable challenge in evaluating texture optimization. With close but inexact geometry and alignment, our method can still produce textures that are consistent with the image queries due to NOC-guided optimization, while baseline methods struggle to transfer textures from incomplete and cluttered real-world objects.

Closely-matching geometry and unknown image alignment. We further consider texturing from a single real-

Figure 7: Texture transfer from real images (ScanNet) to closely matching shape geometry (ShapeNet). Despite inexact geometry and pose, Mesh2Tex produces perceptually consistent texturing.

world image to close shape geometry with unknown image alignment. We show qualitative results in Figure 7 and quantitative evaluation in Tables 4 and 5. Our NOC-guided optimization through our texture manifold generates more perceptually representative textures.

Texture transfer. Finally, we evaluate the challenging texture transfer scenario from a single real-world image from ScanNet [7] to an arbitrary ShapeNet [4] mesh of the same class category. This reflects potential texturing applications from users who wish to use an inspirational image to generate texturing on a given shape. We show a quantitative comparison in Tables 6 and 7, and qualitative results in Figure 8. Our mesh-field texture representation maintains more detail with NOC-guided patch loss to provide more robustness during optimization.

4.3. Ablations

What is the impact of our mesh-field representation for texture generation? Texturify [40] represents a per-face mesh color generation approach, while we introduce a shared barycentric neural field to capture high-fidelity details. Figure 5 shows our much finer-level detail generation, particularly for complex textures on cars.

What is the impact of patch style loss in the texture optimization process? Table 8 shows that our patch-based

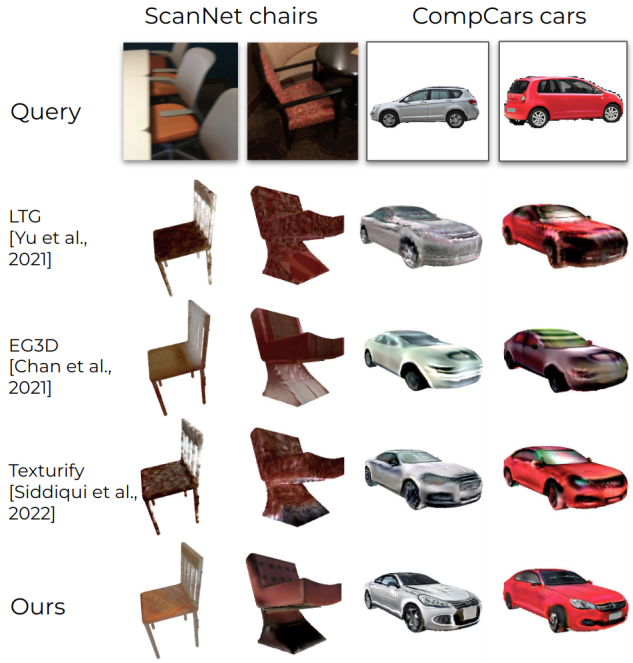

Figure 8: Texture transfer from real images (ScanNet, CompCars) to arbitrary shape geometry of the same class category (ShapeNet). Under this challenging scenario, our NOC-guided patch optimization enables plausible texturing for image queries.

Method	Chairs ShapeNet		Cars ShapeNet		Chairs ScanNet	
	Exact geometry				Close geometry	
	FID	KID	FID	KID	FID	KID
EG3D	146.6	9.49	297.9	20.82	318.8	27.39
Yu et al.	153.1	10.2	289.4	18.88	329.8	28.60
GET3D	265.6	26.67	248.8	15.57	336.5	38.74
Texturify	155.9	10.94	301.4	20.63	342.2	33.65
Ours	**115.4**	**5.38**	**165.5**	**6.84**	**309.7**	**26.50**

Table 2: Evaluation on **aligned queried texture generation** with image queries from rendered ShapeNet chairs and cars, as well as real-world images from ScanNet chairs. Mesh2Tex generates more realistic textures from single-image queries. The KID values are scaled by 10^2.

style loss significantly improves texture transfer optimization performance, allowing for capture of more local detail and robustness to mismatches to the query image.

What is the impact of NOC guidance for the patch style loss? In Table 8, we see that the NOC guidance helps to improve the perceptual quality of results by establishing coarse correspondence to the query image.

What is the effect or surface feature optimization? Following latent texture code optimization, we allow for surface features in the texture generator to further be optimized, which produces slightly improved local texture detail, as

Method	Chairs ShapeNet		Cars ShapeNet		Chairs ScanNet
	Exact geometry				Close geometry
	CLIP	LPIPS	CLIP	LPIPS	CLIP
EG3D	89.87	16.41	83.27	15.69	82.86
Yu et al.	89.73	16.26	84.33	14.83	82.93
GET3D	85.00	-	81.38	-	78.52
Texturify	89.22	16.23	83.82	**14.69**	81.71
Ours	**92.27**	**14.64**	**85.91**	14.77	**84.28**

Table 3: Evaluation on **aligned queried texture generation** with image queries from rendered ShapeNet chairs and cars, as well as real-world images from ScanNet chairs. Our texture transfer optimization produces more perceptually representative texturing results.

Method	Chairs ShapeNet		Cars ShapeNet		Chairs ScanNet	
	Exact geometry				Close geometry	
	FID	KID	FID	KID	FID	KID
EG3D	155.2	9.13	326.8	24.48	319.4	27.18
Yu et al.	160.0	9.88	343.3	32.71	327.1	28.20
GET3D	271.9	26.54	254.4	16.63	337.3	38.76
Texturify	174.2	11.74	344.2	25.34	333.3	28.81
Ours	**121.1**	**5.08**	**165.8**	**6.77**	**311.9**	**26.73**

Table 4: Evaluation on **unaligned queried texture generation** with image queries from rendered ShapeNet chairs and cars, as well as real images from ScanNet chairs. Mesh2Tex maintains more realistic texture details than state of the art. The KID values are scaled by 10^2.

Method	Chairs ShapeNet		Cars ShapeNet		Chairs ScanNet
	Exact geometry				Close geometry
	CLIP	LPIPS	CLIP	LPIPS	CLIP
EG3D	89.51	16.83	81.93	16.14	83.40
Yu et al.	88.98	16.81	79.35	21.84	82.67
GET3D	84.66	-	81.54	-	79.02
Texturify	88.24	16.82	80.45	15.42	82.26
Ours	**91.57**	**14.96**	**85.42**	**14.97**	**84.28**

Table 5: Evaluation on **unaligned queried texture generation** to ShapeNet chairs and cars from rendered ShapeNet objects as well as real-world image queries from ScanNet chairs. Our NOC-guided patch style loss maintains more robustness to the unknown image alignment.

shown in Table 8.
We refer to the supplemental for qualitative ablation analysis.

Limitations. Mesh2Tex offers a promising step towards conditional mesh texturing from a single image, though various limitations remain. For instance, our approach does not explicitly model semantics, potentially leading to distortions in texture in semantically meaningful areas (e.g., spokes of a car wheel). Additionally, a more explicit characterization of the texture distribution could enable probabilistic sampling

Method	Chairs ShapeNet		Cars ShapeNet		Chairs ScanNet		Cars CompCars	
	FID	KID	FID	KID	FID	KID	FID	KID
EG3D	297.4	28.53	343.6	26.57	355.2	33.31	352.3	39.15
Yu et al.	303.4	29.10	351.1	26.75	362.6	34.20	385.4	51.37
GET3D	-	-	-	-	-	-	-	-
Texturify	307.9	29.63	357.1	27.05	370.4	34.85	264.7	29.23
Ours	**276.7**	**26.17**	**194.5**	**10.11**	**348.5**	**32.33**	**203.7**	**21.48**

Table 6: Evaluation on **texture transfer** to ShapeNet chairs and cars from rendered ShapeNet objects as well as real-world image queries from ScanNet chairs and CompCars cars. In this challenging scenario, our mesh-field representation retains more realistic detail in texture optimization. The KID values are scaled by 10^2.

Method	Chairs ShapeNet	Cars ShapeNet	Chairs ScanNet	Cars CompCars
	CLIP	CLIP	CLIP	CLIP
EG3D	81.62	80.72	81.54	68.18
Yu et al.	81.31	78.69	81.21	64.76
GET3D	-	-	-	-
Texturify	80.70	78.59	80.99	66.93
Ours	**83.47**	**82.20**	**81.95**	**77.71**

Table 7: Evaluation on **texture transfer** to ShapeNet chairs and cars from rendered ShapeNet objects as well as real-world image queries from ScanNet chairs and CompCars cars. Our NOC-guided patch style loss enables better characterization of the input query image for texturing.

Method	Chairs (unaligned) ShapeNet		
	FID	KID	CLIP
w/o NOC	124.3	5.53	91.09
w/o patches	143.5	6.91	89.68
w/o surf. features	127.1	5.80	90.70
Ours	**121.1**	**5.08**	**91.57**

Table 8: **Ablation study** on texture transfer optimization design. Our NOC guidance, patch style loss, and surface feature optimization help to improve generated texture quality. The KID values are scaled by 10^2.

when input queries may be occluded or incomplete.

5. Conclusion

We presented Mesh2Tex, which learns a texture manifold based on a hybrid mesh-field representation to generate realistic, high-quality textures on shape geometry from real-world imagery. Mesh2Tex enables test-time optimization for texture transfer from single image queries. Crucially, our NOC-guided local and global style loss enables optimization for perceptually matching textures on a 3D shape that does not require exact geometric matches to RGB image queries. We believe this opens up many new possibilities in texturing for content creation, enabling easy-to-capture images to be used as guidance for holistic shape texturing.

Acknowledgements. This work was supported by the Bavarian State Ministry of Science and the Arts coordinated by the Bavarian Research Institute for Digital Transformation (bidt). ST was partially supported by a gift award from CISCO.

References

[1] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning CAD model alignment in RGB-D scans. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2614–2623, 2019. 5

[2] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 5

[3] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *arXiv*, 2021. 2, 5

[4] 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. 5, 7, 11

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[6] Manuel Dahnert, Ji Hou, Matthias Nießner, and Angela Dai. Panoptic 3d scene reconstruction from a single rgb image. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2

[7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5, 7

[8] Angela Dai, Yawar Siddiqui, Justus Thies, Julien Valentin, and Matthias Nießner. Spsg: Self-supervised photometric scene generation from rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2021. 2

[9] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *IEEE Computer Vision and Pattern Recognition*, 2021. 2

[10] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *Advances In Neural Information Processing Systems*, 2022. 2, 5

[11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 4

[12] 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. 4, 17

[13] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. Leveraging 2d data to learn textured 3d mesh generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7498–7507, 2020. 2

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 5

[15] Jingwei Huang, Justus Thies, Angela Dai, Abhijit Kundu, Chiyu Jiang, Leonidas J Guibas, Matthias Niessner, and Thomas Funkhouser. Adversarial texture optimization from rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1559–1568, 2020. 2

[16] Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J Guibas. Quadriflow: A scalable and robust method for quadrangulation. In *Computer Graphics Forum*, volume 37, pages 147–160. Wiley Online Library, 2018. 3

[17] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *CVPR*, 2022. 3

[18] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023. 3

[19] Xie Jianwen, Yifei Xu, Zilong Zheng, Song Zhu, and Yingnian Wu. Generative pointnet: Energy-based learning on unordered point sets for 3d generation, reconstruction and classification. 04 2020. 2

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *CVPR*, 2020. 3

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[22] Juil Koo, Ian Huang, Panos Achlioptas, Leonidas J Guibas, and Minhyuk Sung. Partglot: Learning shape part segmentation from language reference games. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[23] Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Patch2cad: Patchwise embedding learning for in-the-wild shape retrieval from a single image. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12569–12579, 2021. 2

[24] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 2

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 2

[26] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *arXiv preprint arXiv:2112.03221*, 2021. 3

[27] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. AutoSDF: Shape priors for 3d completion, reconstruction and generation. In *CVPR*, 2022. 2

[28] Charlie Nash, Yaroslav Ganin, S. Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. 02 2020. 2

[29] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019. 2

[30] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[31] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 2

[32] 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, 17

[33] Dario Pavllo, Jonas Kohler, Thomas Hofmann, and Aurelien Lucchi. Learning generative models of textured 3d meshes from real-world images. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2, 5

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

[36] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[37] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[38] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2020. 2

[39] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-forge: Towards zero-shot text-to-shape generation. *arXiv preprint arXiv:2110.02624*, 2021. 3

[40] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces, 2022. 2, 3, 4, 5, 7, 14

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 17

[42] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 4, 17

[43] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. *ArXiv*, abs/2112.05139, 2021. 3

[44] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 3, 4

[45] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 492–502. Curran Associates, Inc., 2019. 2

[46] 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. 5, 17

[47] Rui Yu, Yue Dong, Pieter Peers, and Xin Tong. Learning texture generators for 3d shape collections from internet photo sets. In *British Machine Vision Conference*, 2021. 5

[48] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[49] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics*, 33, 08 2014. 2

In this supplemental material, we show additional texture generation results in Section A, texture transfer from query images in Section B, qualitative visualizations for ablations in Section C, network architecture details in Section D, and additional implementation details in Section E.

A. Additional Generation Results

In Fig. 9 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.

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

In Figs. 10, 11, and 12, 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. 13, 14, and 15 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.

C. 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.

Figure 9: Additional results on unconditional texturing for meshes from ShapeNet [4], in comparison with state of the art. Our approach generates more realistic, detailed textures.

Figure 10: Optimized textures based on input query images (top row) using aligned query images from ShapeNet chairs and cars.

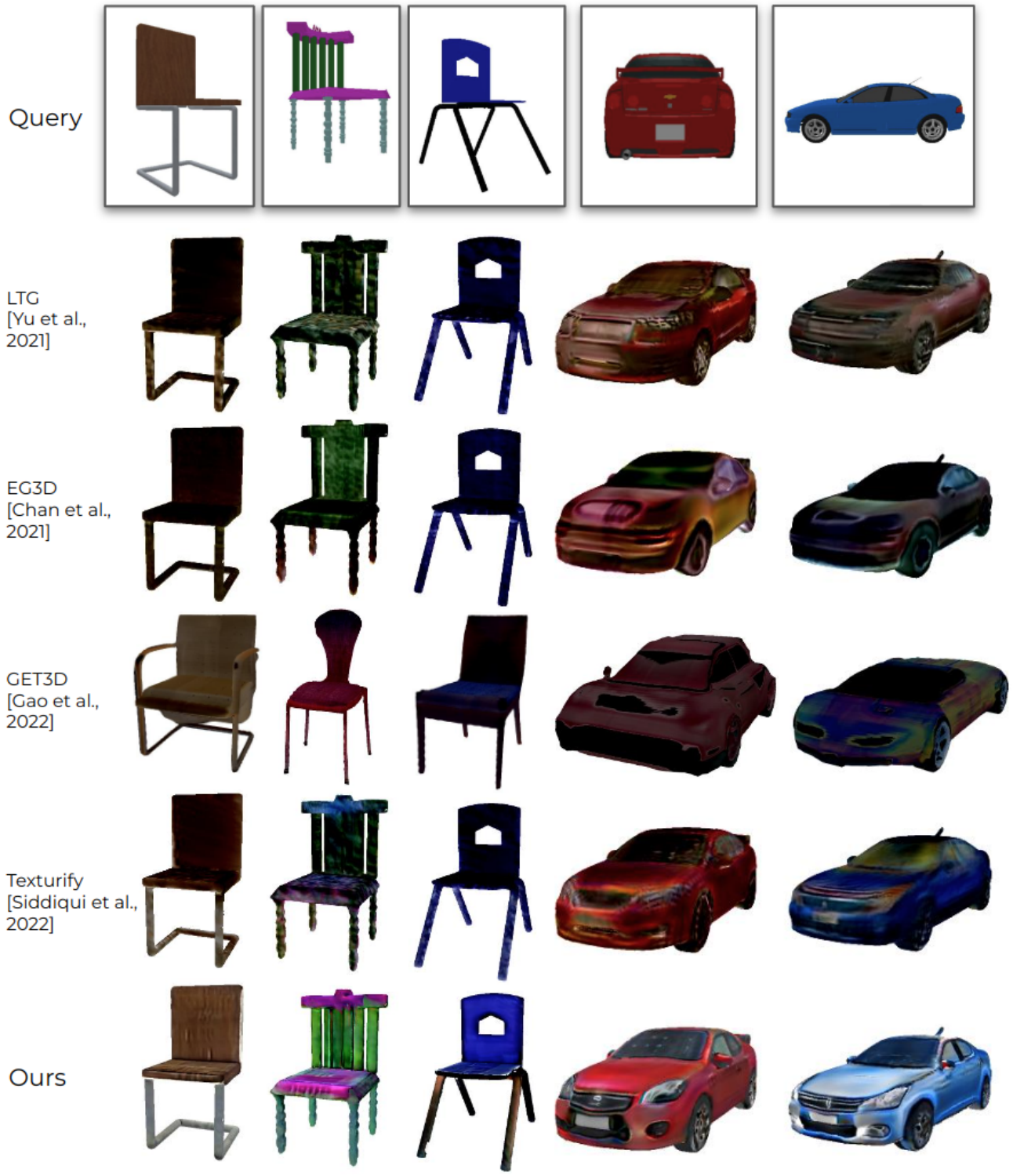

Figure 11: Optimized textures based on input query images (top row) using unaligned images query images from ShapeNet chairs and cars.

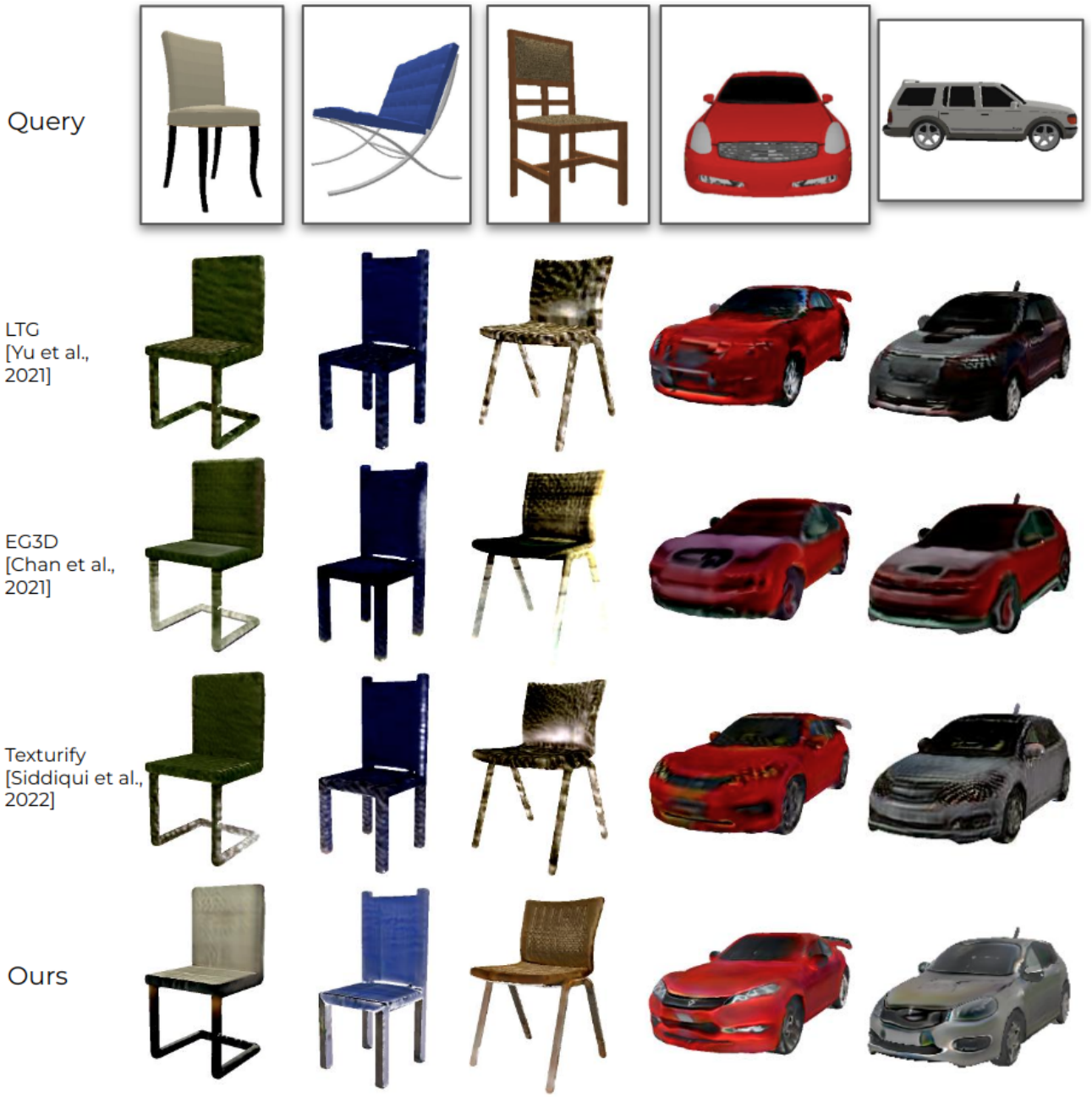

Figure 12: Texture transfer from input query images (top row) using unaligned images and arbitrary shape geometry of the same class category (ShapeNet).

D. Network Architecture

An overview of the architectures of our surface encoder, generator, and neural field Ψ is shown in Figs. 17 and 18. Note that we follow the use of FaceResNet blocks, FaceConv layers, and Synthesis Block from Texturify [40]. 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.

Figure 13: Texture transfer from real-world input query images (top row, ScanNet) using aligned images and close shape geometry (ShapeNet).

Figure 14: Texture transfer from real-world input query images (top row, ScanNet) using unaligned images and similar shape geometry (ShapeNet).

Figure 15: 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).

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 [12] 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 [42] 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.

E. 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 [32], and learning rates of 1e-4, 15e-4, 5e-4, 1e-4 for CompCars [46]. 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 [41] network,

Figure 16: Qualitative ablation study on texture transfer from synthetic input query images (ShapeNet) using unaligned images; visualized with the query image pose.

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).

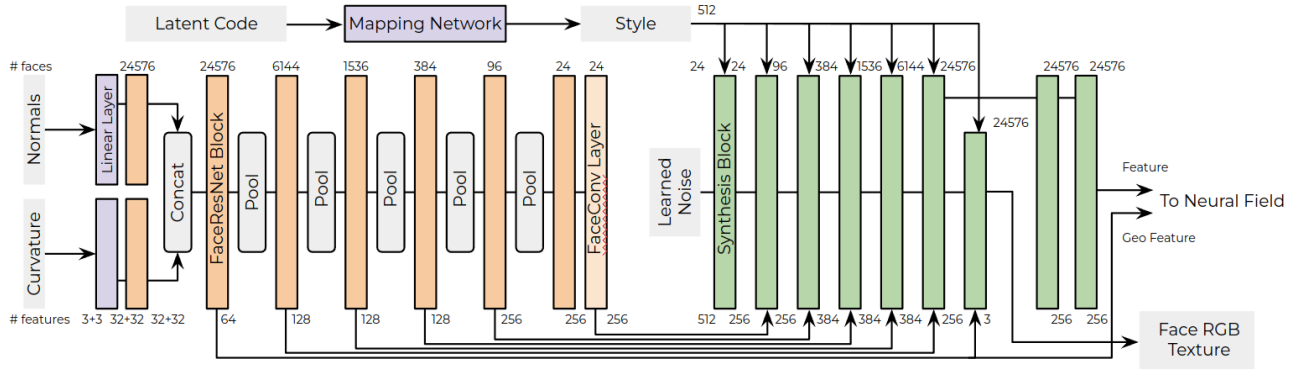

Figure 17: Overview of Surface Encoder (orange) and Generator (green) architectures. The encoder takes as input normals and curvatures of the finest resolution quadmesh, processes them in a hierarchical convolutional structure to extract geometric features. The generator then considers a latent texture code, learned noise, and the geometric features to produce per-face features for the neural field.

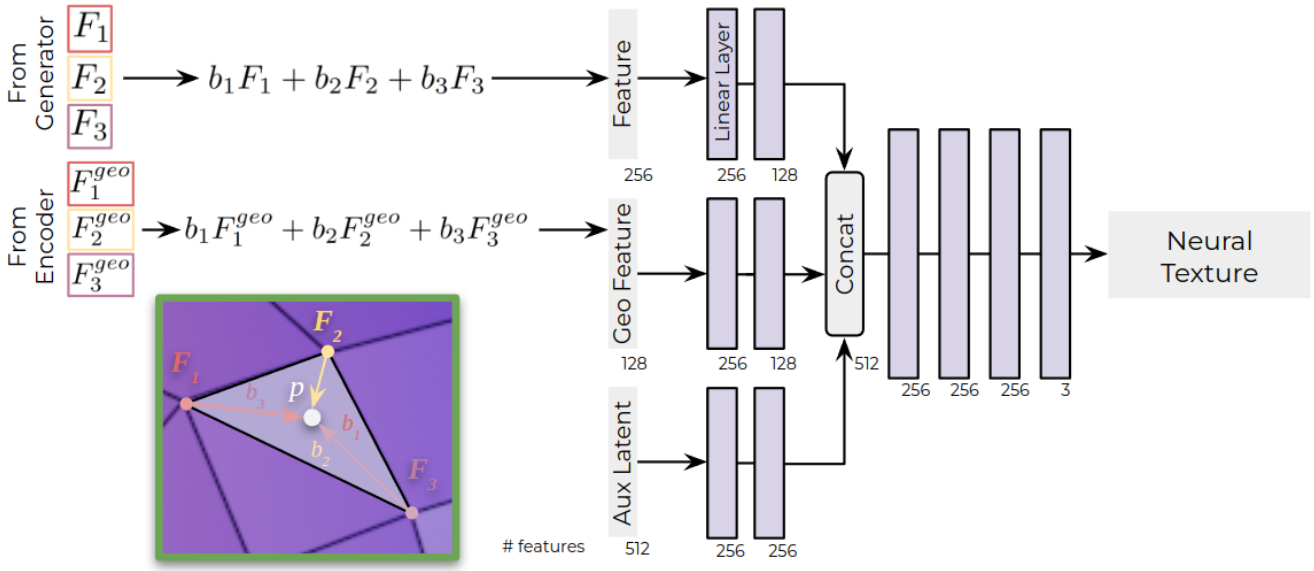

Figure 18: Our architecture for local face neural field Ψ. Features from the surface encoder and generator are fused by their barycentric coordinates and passed with an auxiliary latent vector into an MLP to produce the final color of surface location p.