StyleMesh: Style Transfer for Indoor 3D Scene Reconstructions

Lukas Höllein[1] Justin Johnson[2] Matthias Nießner[1]
[1]Technical University of Munich [2]University of Michigan

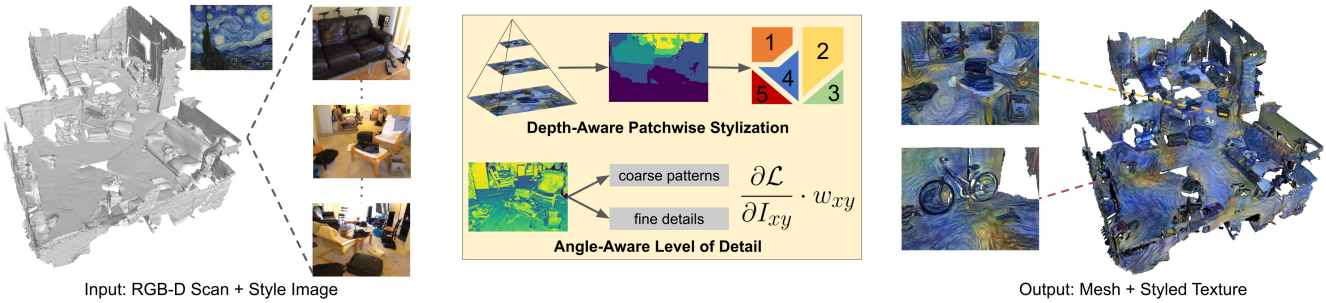


Figure 1. We perform style transfer on reconstructed 3D meshes by synthesizing stylized textures. We compute style transfer losses on views of the scene and backpropagate gradients to the texture. Depth and surface normal data from the mesh enable 3D-aware stylization, preventing artifacts that arise from standard 2D losses. Our stylized meshes can be rendered using the traditional graphics pipeline.

Abstract

We apply style transfer on mesh reconstructions of indoor scenes. This enables VR applications like experiencing 3D environments painted in the style of a favorite artist. Style transfer typically operates on 2D images, making stylization of a mesh challenging. When optimized over a variety of poses, stylization patterns become stretched out and inconsistent in size. On the other hand, model-based 3D style transfer methods exist that allow stylization from a sparse set of images, but they require a network at inference time. To this end, we optimize an explicit texture for the reconstructed mesh of a scene and stylize it jointly from all available input images. Our depth- and angle-aware optimization leverages surface normal and depth data of the underlying mesh to create a uniform and consistent stylization for the whole scene. Our experiments show that our method creates sharp and detailed results for the complete scene without view-dependent artifacts. Through extensive ablation studies, we show that the proposed 3D awareness enables style transfer to be applied to the 3D domain of a mesh. Our method [1] can be used to render a stylized mesh in real-time with traditional rendering pipelines.

1. Introduction

Creating 3D content from RGB-D scans is a popular topic in computer vision [1, 12, 29, 43, 44]. We tackle a novel use case in this area: stylization of a reconstructed mesh with an explicit RGB texture. Neural Style Transfer (NST) shows great results for stylization of images or videos, but stylization of 3D content like meshes has been underexplored. We synthesize a texture for the mesh which is a combination of observed RGB colors and a painting's artistic style. After stylization, one could explore the space in VR and see it painted in the style of Van Gogh.

Our use case is similar to prior texture mapping methods [2,16,26,27,29,53,57] which construct a texture from a set of posed RGB images, but we produce a stylized texture rather than directly matching input images. This is difficult since style transfer losses are typically defined on 2D image features [20], so NST does not immediately generalize to 3D meshes. Recently, style transfer has been combined with novel view synthesis to stylize arbitrary scenes with a neural renderer from a sparse set of input images [7,25,35]. These model-based methods require a forward pass during inference and cannot directly be applied to meshes. Kato et al. [32] and Mordvintsev et al. [42] use differentiable rendering to bridge the gap between image style transfer and texture mapping: backpropagating image losses to a texture representation enables consistent mesh stylization.

However, applying these methods to room-scale geometry is challenging as the resulting stylization patterns are noisy and can contain view-dependent stretch and size artifacts. For example, optimizing a surface from a small grazing angle creates patterns in the image plane for that pose. Viewing the same surface from an orthogonal angle then

[1] https://lukashoel.github.io/stylemesh/

shows stretched-out patterns due to the perspective distortion. Similarly, seeing an object from close and far-away viewpoints mixes small and large patterns on the same surface. Perceiving the depth thus becomes harder, due to inconsistent stylization sizes. These issues arise because 2D style transfer losses do not incorporate 3D data like surface normals and depth. Instead, textures are separately stylized in each pose's image plane.

To this end, we formulate an energy minimization problem over the texture that combines texture mapping with style transfer (similar to [42]) and minimizes style transfer losses for each pose in a 3D-aware manner that avoids view-dependent artifacts. First, we utilize depth to render image patches at increasingly larger screen-space resolutions. By splitting the style loss calculation over these patches, we create larger stylization patterns in the foreground than the background. As a result, patterns have the same size in world-space and are optimized in a view-independent way. Second, we use the angle between the surface normal and view direction to determine the degree of stylization for each pixel. By calculating Gram matrices from different style image resolutions (similar to [39]) areas seen from small grazing angles are stylized with coarse details, which are later refined if they are seen from better angles. Third, we avoid discretization artifacts by scaling gradients with per-pixel angle and depth weights during backpropagation.

Compared to state-of-the-art 3D style transfer methods, our experiments show an improvement in terms of 3D consistent stylization both qualitatively and quantitatively. Additionally, our explicit texture representation allows for direct usage with traditional rendering pipelines.

To summarize, our contributions are:

- Style transfer for room-scale indoor scene meshes with a new texture optimization, which results in 3D consistent textures and mitigates view-dependent artifacts.
- A depth-aware optimization at different screen-space resolutions, that creates equally-sized stylization patterns in the world-space of the mesh.
- An angle-aware optimization at different stylization details, that creates unstretched stylization patterns in the world-space of the mesh.

2. Related Work

Our approach is a NST method operating on the texture parametrization of a mesh. It is related to recent work on style transfer for videos and 3D objects, as well as texture generation from RGB-D images.

Texture Mapping. Many methods texture a reconstructed mesh from multiple RGB images, i.e., they map a texture onto the geometry that combines the color information of all images [2,16,26,27,29,53,57]. These methods must handle inaccuracies in pose, geometry, color and distortions to find the best texture for the scene. In contrast, we aim to create a texture that is also styled to a specific image and avoid view-dependent stylization artifacts by introducing depth- and angle-awareness into the optimization.

Image Style Transfer. NST, first introduced in Gatys et al. [20], can be optimization-based [8, 20, 21] or model-based [15, 28, 30, 52]. It is inherently defined in the image domain by matching CNN features either globally or in a local, patch-based manner [20, 31, 34, 36, 41]. Thus, it cannot directly utilize 3D data like depth or surface normals of a mesh. This can lead to view-dependent stylization artifacts when optimizing a texture through multiple poses. We induce 3D-awareness into optimization-based NST by splitting the loss calculation across different image segments.

Video Style Transfer. Video style transfer (VST) methods consistently stylize RGB video frames with a given style. These methods are optimization-based [46, 47] or model-based [5, 6, 18, 19, 22, 54, 55] and employ temporal consistency or optical flow constraints. Other methods combine features in a temporally consistent way, without using optical flow or depth constraints directly [15,37]. VST methods can be combined with texture mapping to achieve consistent stylization of indoor scenes. However, since VST optimizations are unaware of the underlying 3D structures, the resulting textures are often blurry or low-detail.

3D Style Transfer. Lifting style transfer into 3D has been explored for texturing individual objects [32, 42, 56] or faces [23]. However, they focus on isolated objects (not room-scale scenes) and do not utilize 3D data. In contrast, our method stylizes complete indoor scenes in a 3D-aware way. Another line of work applies exemplar-based NST to 3D models [24, 49], guiding the stylization process explicitly from (hand-crafted) examples. In contrast, we follow original NST by stylizing 3D scene models from artistic paintings and camera images. Cao et al. [3] stylize indoor scenes using a point cloud that cannot be directly used to texture a mesh. Other methods combine novel view synthesis and NST for consistent stylization from only a few input images [7, 25, 35]. In contrast, we do not require a network during inference to produce stylization results; our results can be rendered by a standard graphics pipeline.

3. Method

Our goal is to stylize the mesh of an indoor scene: we want to create a texture that is a mixture of original RGB colors and a style image. To avoid view-dependent artifacts, we formulate a depth- and angle-aware optimization problem over all images. We require a set of N images $\{I_k\}_{k=1}^{N}$ captured at different poses. We also need a mesh reconstruction of the scene for which we create a texture parametrization, i.e., we need a uv coordinate per vertex.

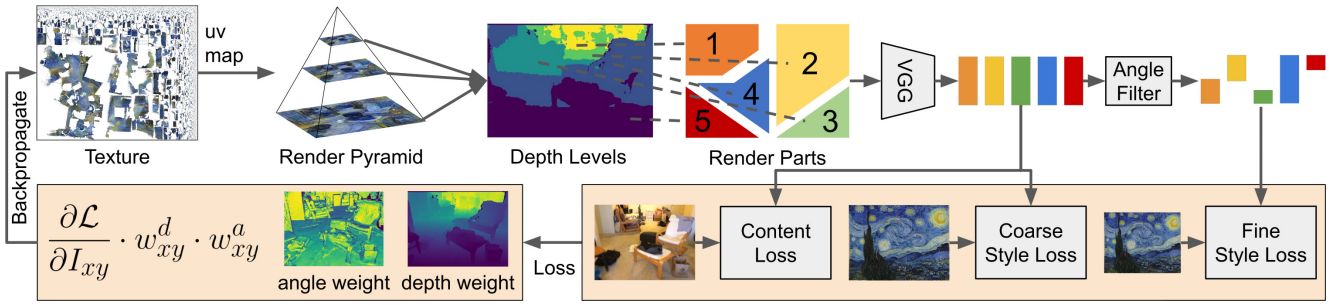


Figure 2. We optimize a texture for the mesh reconstruction of a scene using multiple RGB images and a style image. We sample the texture with uv maps at different resolutions, yielding a render pyramid for each pose. Using depth, we divide the screen space into parts, each corresponding to a different pyramid level. Each is encoded separately and compared with its RGB image part in the content loss. Using normals, the style loss is further split into fine and coarse branches. We discard features seen from small grazing angles for fine stylization. Finally, we scale the resulting image gradients with continuous angle and depth weights before backpropagating to the texture.

For each pose, we sample the texture with the corresponding uv map at multiple resolutions, yielding a render pyramid. Depending on the depth of each pixel, we split the image into multiple render parts, each belonging to one pyramid resolution. Each part is used in content and style losses, where we only stylize pixels with fine details, that are seen from good angles. Finally, we smooth the per-pixel gradients before backpropagating to the texture. The complete method is visualized in Fig. 2.

3.1. Texture Optimization

We optimize a stylized RGB texture $\mathcal{T}^*$ from all RGB images $\{I_k\}_{k=1}^N$ and a separate style image I_s. Similar to [42], we formulate a minimization problem with content and style losses $\mathcal{L}_c, \mathcal{L}_s$ and add a regularization term $\mathcal{L}_r$:

$$\mathcal{T}^* = \arg\min_{\mathcal{T}} \sum_{i=0}^{N} (\lambda_c \mathcal{L}_c(I_i, \hat{P}_i) + \lambda_s \mathcal{L}_s(I_s, \hat{P}_i) + \lambda_r \mathcal{L}_r(\mathcal{T})) \tag{1}$$

where $\hat{P}_i$ is the render pyramid for the current pose, sampled from the texture with the corresponding uv maps and $\lambda_c, \lambda_s, \lambda_r$ are loss weights. The sampling operation is identical to traditional graphics and differentiable, i.e., we bilinearly interpolate each pixel from four neighboring texels. Similar to Thies et al. [51], we define our texture using a Laplacian Pyramid to regularize the texels in each layer with $\mathcal{L}_r$. This helps avoid magnification and minification artifacts and reduces visible noise in the texture. For each pose, we optimize the subset of observed texels. Thus, we require a pose set covering most of the scene to optimize the texture completely. In contrast, stylizing in texture space directly [56] is problematic for a room-scale texture parametrization, which may contain many seams.

3.2. Depth Level Render Parts

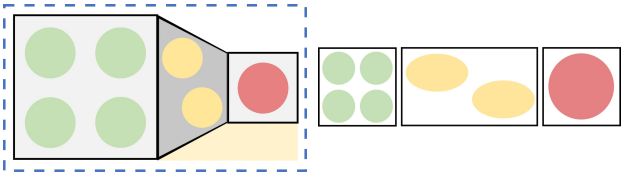
Style transfer operates on the CNN features of an image [20]. This leads to a limited sense of depth when optimizing over multiple poses. Stylization patterns can appear equally large in the foreground and background, e.g., when parts of a surface are seen far-away and close-up (see Fig. 3). Observing the same surface from multiple poses thus mixes small and large patterns next to each other. As a result, renderings using the optimized texture do not convincingly capture depth. Liu et al. [38] make style transfer depth-aware in the image plane with a depth-loss network. In contrast, we incorporate depth-awareness by optimizing at multiple screen-space resolutions. Larger patterns appear in the foreground than the background of an image, ultimately leading to equally large style in world space.

We make use of the relation that area in screen-space is inversely proportional to depth, i.e., when depth increases by a factor of p, a given projected area decreases by p^2. On the other hand, style transfer is agnostic to the image resolution that it is applied on, i.e., when resolution increases by p^2, stylization patterns appear proportionally smaller (because the receptive field becomes smaller relative to the resolution) [21]. We combine both relations to optimize stylization patterns having the same size in world space: when depth increases by p, we increase image resolution by p^2.

We apply the relation to divide the image into parts, sampled from the render pyramid at increasingly larger resolutions. The content and style losses are then calculated independently for each part. To discretize into parts, we define a minimum depth value θ_d, making the relation absolute. We calculate the optimal image height per-pixel as

$$R_{xy} = \theta_{min} \cdot \frac{d_{xy}}{\theta_d} \tag{2}$$

where d_{xy} is the depth at pixel (x, y) and θ_{min} is the minimum resolution. We express resolution R_{xy} as height in pixels and scale the width accordingly. We then map R_{xy} to the nearest neighbor in the render pyramid, yielding its index as depth level per-pixel. Finally, we apply a 3×3 erosion kernel to smooth the depth level map over all pixels.



(a) Projected Geometry (b) Geometry in World-Space

Figure 3. Stylizing mesh faces in 2D (screen-space) is dependent on angle and size of the projected geometry. Optimizing the faces of a wall from a small grazing angle leads to stretched-out stylization patterns in world-space (yellow). Regions of similar size can receive more or less patterns, depending on their projected size (green and red). We denote stylization patterns as circles that are optimized from screen-space into texture space.

3.3. Angle Filter

Style transfer in screen-space can create stretched-out stylization patterns (see Fig. 3). Patterns might look circular from one view, but are stretched-out ellipses in world space (e.g., when optimizing from a small grazing angle). To prevent this, we combine coarse and fine style losses and optimize fine details only for areas seen from good angles. Similar to previous work [21,39], we utilize the fact that the receptive field of high-resolution images is still small [40]. As a result, stylization patterns appear coarser and less detailed, when optimized from a larger style image. We find that coarse patterns are less prone to stretch artifacts.

For each pixel, we calculate its normal-to-view angle $\alpha_{xy} = \measuredangle(\vec{n}_{xy}, \vec{v})$ where $\vec{n}_{xy}$ is the interpolated surface normal at pixel (x, y) and $\vec{v}$ is the viewing direction. Only the pixels where $\alpha_{xy} \leq \theta_a$ are used for the style loss with a low-resolution style image that produces fine stylizations. We always use all pixels and a high-resolution style image to optimize coarse stylization patterns. This creates a combination of coarse and fine patterns without stretch artifacts.

3.4. Multi-Resolution Part-based Losses

Multiple content and style losses combine depth levels (Sec. 3.2) and angle filtering (Sec. 3.3) to optimize the texture without view-dependent artifacts. We encode the render pyramid $\hat{P}$ with a pretrained VGG network [48] into the feature pyramid $\hat{F}$. Using the depth level map, we only keep corresponding features in each layer of $\hat{F}$. We compute a coarse Gram matrix $\hat{G}_c$ and an angle-filtered fine one $\hat{G}_f$ from the features in every layer. Similarly, G_c and G_f correspond to the high- and low-resolution style images. We define the style loss as

$$\mathcal{L}_s(I_s, \hat{P}) = \sum_l^{\theta_l} \hat{w}_l \cdot (||G_c - \hat{G}_c^l||_F^2 + ||G_f - \hat{G}_f^l||_F^2) \quad (3)$$

which sums over all depth levels independently (part-based) and combines coarse and fine stylization (multi-resolution).

We calculate the normalized weighting factor $\hat{w}_l$ as

$$w_l = \frac{v_l}{t_l} \quad \text{and} \quad \hat{w}_l = \frac{w_l}{\sum_i^{\theta_l} w_i} \quad (4)$$

where v_l is the visible and t_l the total number of pixels in depth level l. Similarly, the content loss is defined as

$$L_c(I, \hat{P}) = \sum_l^{\theta_l} \hat{w}_l \cdot ||F^l - \hat{F}^l||_F^2 \quad (5)$$

where F^l are the features of the content image I, split in a similar way. For brevity, we omit different VGG layers and image indices from the notation. As proposed in Gatys et al. [20], we use the layers *relu_{1-5}_1* for the style loss and *relu_4_2* for the content loss. We calculate the losses independently for every VGG layer and sum them accordingly.

3.5. Per-Pixel Gradient Scaling

Depth levels (Sec. 3.2) and angle filtering (Sec. 3.3) impose hard thresholds on the image of each pose. To avoid discretization artifacts at decision boundaries, we scale the per-pixel gradients before backpropagating them to the texture. First, we calculate a weighting factor $w_{xy}^a = cos(\alpha_{xy})$ from the normal-to-view angle α_{xy}. This controls the influence of a pose on each pixel by preferring orthogonal over small grazing viewing angles. Scaling features similar to Gatys et al. [21] instead results in oversaturation artifacts.

Second, we adapt the idea of trilinear Mipmap interpolation [17]. Each pixel contributes to the render parts of its nearest two pyramid layers, resulting in two per-pixel gradients. We calculate the distance to the nearest layer as

$$w_{xy}^d = |\frac{R_{xy} - L_{xy}^1}{L_{xy}^1 - L_{xy}^2}| \quad (6)$$

where R_{xy} is the optimal resolution for pixel (x, y) and L_{xy}^1, L_{xy}^2 are the resolutions of the nearest and second nearest pyramid layers. Finally, we linearly interpolate between the per-pixel gradients as

$$\frac{\partial \mathcal{L}}{\partial I_{xy}} = \frac{\partial \mathcal{L}_1}{\partial I_{xy}} \cdot w_{xy}^d \cdot w_{xy}^a + \frac{\partial \mathcal{L}_2}{\partial I_{xy}} \cdot (1 - w_{xy}^d) \cdot w_{xy}^a \quad (7)$$

where $\mathcal{L}_1$ is the loss term for the nearest pyramid layer of pixel (x, y) and $\mathcal{L}_2$ for the second nearest, respectively.

3.6. Data Preprocessing

We use the ScanNet [10] and Matterport3D [4] datasets, which provide RGB-D images and reconstructed meshes (we use per-region meshes for Matterport3D [4]). We use the RGB images for optimization, but filter them with a Laplacian kernel to remove blurry images. We reduce each mesh's complexity by merging vertices until $\leq$ 500K faces remain. Then, we generate a texture parametrization with Blender's smart uv project [9] with an angle limit of 70°. We precompute the uv maps for each estimated pose.

4. Results

Implementation Details. We optimize textures at a resolution of 4096×4096 as a Laplacian Pyramid [51] with 4 layers and regularization strength λ_r=5000. We use λ_c=70 and λ_s=0.0001 for content and style loss weights. We optimize for 7 epochs and repeat each frame 10 times. We set θ_{min}=32 and use θ_l=4 render pyramid layers at heights of $\{256, 432, 608, 784\}$ pixels. We set θ_a=30°, θ_d=0.25 meters for ScanNet [10] and θ_a=40°, θ_d=0.2 meters for Matterport3D [4]. We incrementally halve the original style image resolution until either the width or height reaches a size of 256 pixels. We use the resulting image for the stylization of fine details and a two steps larger image for coarse details. We use Adam [33] with batch size 1 and initial learning rate 1 which decays multiplicatively by 0.1 every 3 epochs. We tried L-BFGS [58] which gave similar results. After optimization, we export the Laplacian Pyramid to a single texture image and use a standard rasterizer with Mipmaps and shading [17] for rendering.

Evaluation Metrics. We conduct a user study to show the advantages of depth- and angle-awareness (Fig. 10). Additionally, we quantify them by stylizing with a "circle" image (Fig. 8). We calculate the correlation between circle size and depth in screen-space (Corr. 2D) and world-space (Corr. 3D), as well as circle stretch as the ratio of horizontal and vertical radius in world-space (Tab. 2). For quantifying 3D consistency, we calculate the L_1 distance between source and reprojected target frames (Tab. 1). Please refer to the supplemental material for more details about metrics.

4.1. Style Transfer on Scenes

Our method competes with 3D style transfer methods that stylize a scene through an explicit or implicit representation. Specifically, we compare our method with DIP of Mordvintsev et al. [42] and NMR of Kato et al. [32]: like us they also optimize a texture, but they do not utilize angle or depth data. Additionally, we compare with LSNV of Huang et al. [25], which uses a neural renderer to stylize point clouds. We show results on the Matterport3D [4] dataset in Fig. 5 and on the ScanNet [10] dataset in Fig. 6. A visualization of textured meshes is given in Fig. 4. Please see the supplemental material for more examples.

Our results show that we are able to stylize scenes without view-dependent size or stretch artifacts. In contrast to the other methods, our approach creates sharp and detailed effects for the complete scene. Optimizing the complete texture is especially difficult for DIP [42] and NMR [32], which both contain noisy texels. LSNV [25] stylizes complete images, but their results are less detailed. To quantitatively evaluate our method and the related approaches, we compute the mean L_1 distance between source frame and a reprojected target frame. The results are listed in Tab. 1.

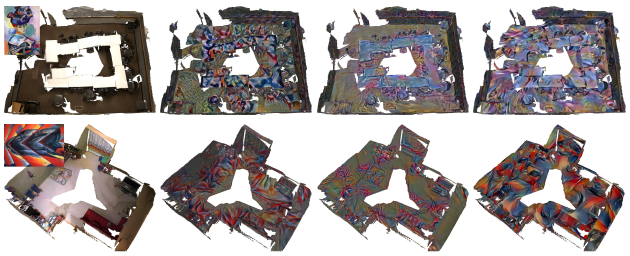


Figure 4. Top-down view on stylized meshes in comparison to previous work.

Method	Short-Range ↓	Long-Range ↓
LSNV [25]	4.873	7.207
NMR [32]	1.565	2.165
DIP [42]	1.396	1.723
Ours	**1.225**	**1.566**

Table 1. L_1 distance between source frame and a reprojected target frame. We report mean values for short-range (2nd next frame) and long-range (20th next frame) in 10 different ScanNet [10] scenes.

4.2. Ablation Studies

Qualitative Comparison. Our method uses per-pixel angle and depth as input to optimize the texture in a 3D-aware manner. This helps avoid view-dependent stretch and size artifacts being optimized into the texture from different poses. We compare only using angle input (no render pyramid) and not using angle/depth (only 2D texture optimization with Laplacian Pyramid representation). In Fig. 7 we can see that using angle makes it easier to distinguish between surfaces like the wall and sofa in row 2. Adding depth creates smaller and detailed patterns in the background (e.g., the strokes in the background of row 1). Please see the supplemental material for more examples.

We optimize all ablation modes such that stylization patterns are equally strong, i.e., style should be similar for a fair comparison. A too low degree of stylization would reduce view-dependent artifacts because original RGB colors get more dominant. Similarly, a too high degree discards content features too much, which increases artifacts.

Quantitative Comparison. We measure the effects of angle- and depth-awareness as follows. We stylize a scene with a "circle" image using only the style loss (see Fig. 8). We then detect ellipses in the resulting images and measure their horizontal and vertical axis lengths. Naturally, NST creates ellipses of different shapes, but their overall distribution reveals the degree of 3D awareness for the complete scene. Inverse correlation between per-pixel depth and ellipse size in screen-space (Corr. 2D) indicates that stylized features are smaller in the background. A weak correlation

Figure 5. Comparison of stylization results for our method and related work on the Matterport3D [4] dataset. We texture the mesh with each method (point cloud for Huang et al. [25] respectively) and render a single pose that is also captured in the RGB images.

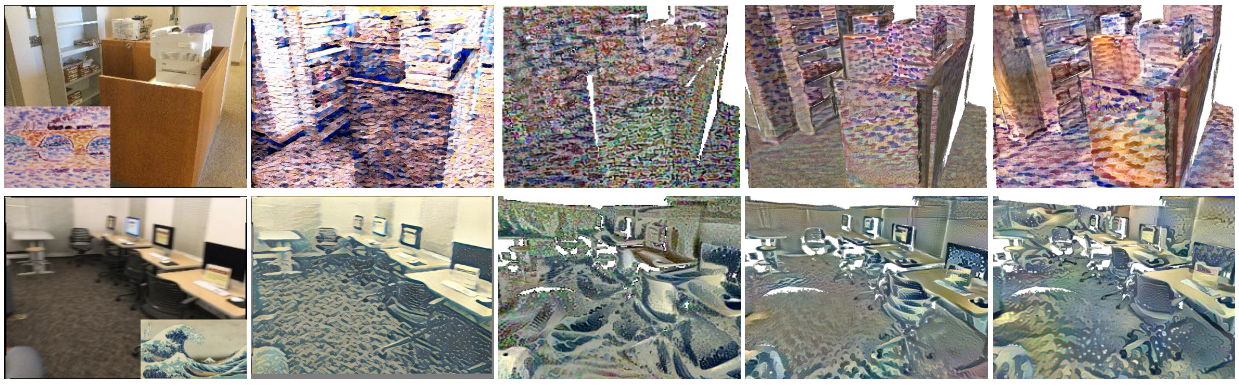


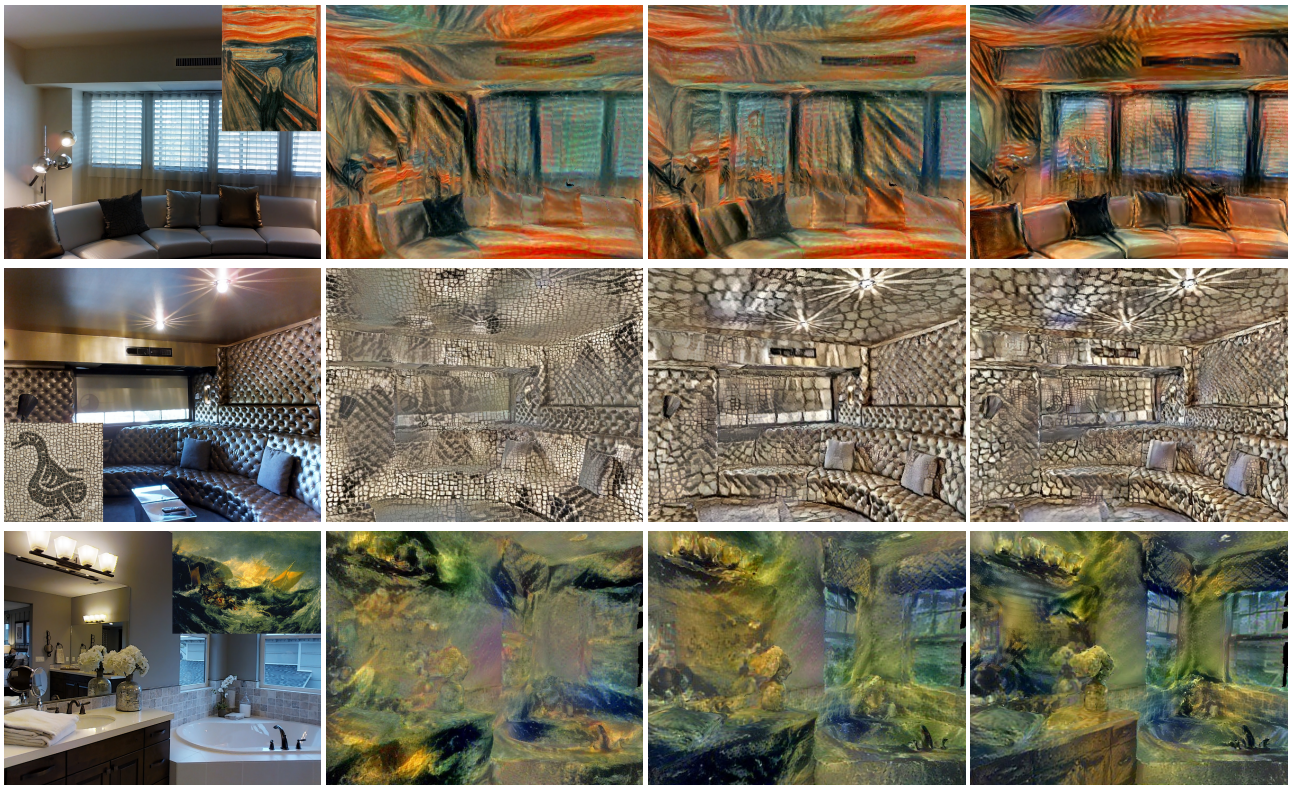
Figure 6. Comparison of stylization results for our method and related work on the ScanNet [10] dataset. We texture the mesh with each method (point cloud for Huang et al. [25] respectively) and render a single pose that is also captured in the RGB images.

in world-space (Corr. 3D) indicates that absolute size is independent of the observed poses. Both metrics together classify the depth-awareness. View-dependent stretch is larger if ellipse's horizontal and vertical axes are of different lengths. The stylization is angle-aware if the stretch is reduced. We do not measure coarse and fine stylization this way, because the "circle" image contains too few high-resolution features. Please see the supplemental material for more details about metric computation. As can be seen in Tab. 2, using angle and depth improves our method.

Depth Scaling. A key piece of our method is the render pyramid of different image resolutions. By tuning the value of θ_d, we change the threshold of when to sample from the

Method	Corr. 2D ↑	Corr. 3D ↓	Stretch ↓
Only 2D	0.172	0.126	3.512
Angle	0.126	**0.110**	3.396
Angle/Depth	**0.538**	0.125	**3.391**

Table 2. Quantitative results of stylizing a scene with a "circle" image. With depth-awareness (Angle/Depth), depth and size strongly correlate in screen-space (2D), but not in world-space (3D). That is, the stylized circle size is smaller in the background of an image, but uniformly distributed in 3D. Adding angle-awareness reduces the circle stretch (Angle). In contrast, circles show more view-dependent artifacts without depth/angle (Only 2D).



(a) RGB and Style (b) Only 2D (c) With Angle (d) With Angle and Depth

Figure 7. Qualitative ablation study of our method. We compare ours (d) against only using angle (c) and not using angle and depth (b). Using angle better distinguishes surfaces and using depth creates smaller/detailed stylization in the background.

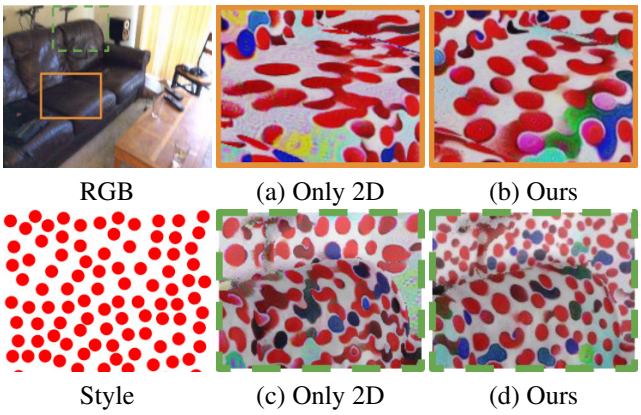


Figure 8. Stylizing with a "circle" image quantifies 3D-awareness. Circles become ellipsoidal on the couch without angle-awareness (a); using angle, they are less distorted (b). Using depth, circles are smaller in the back (d); they are equally large without (c).

next higher resolution. This increases (higher θ_d) or decreases (lower θ_d) the absolute stylization size, while still retaining relative change in size (see Fig. 9). This allows to fine-tune the complete scene until a desired look is obtained.

User Study. We conduct a user study on the effectiveness of our proposed depth- and angle-awareness. Users compared our method against each baseline separately by preferring one of two images. They judged in which image stylization patterns (a) have less visible stretch and (b) are smaller in the background. In total, 20 users each answered 70 questions, comparing against NMR [32], DIP [42] and ours without angle- and depth-awareness (Only 2D). As can be seen in Fig. 10, our method is preferred in both categories.

4.3. Comparison to Video Style Transfer

As an alternative way to optimizing a stylized texture, one could combine video style transfer (VST) methods and RGB texture mapping to produce a stylized scene in two steps (see Fig. 11). We can obtain an RGB texture from all images of the scene and render arbitrary trajectories, that we stylize with a VST method (Tex→VST). However, we never obtain a stylized texture this way and thus need the VST method during inference for each novel pose. Stylization details are also much lower, due to missing details in the RGB texture and reconstructed geometry. By optimizing directly from camera images, we obtain sharper details.

Alternatively, we can stylize a trajectory of camera im-

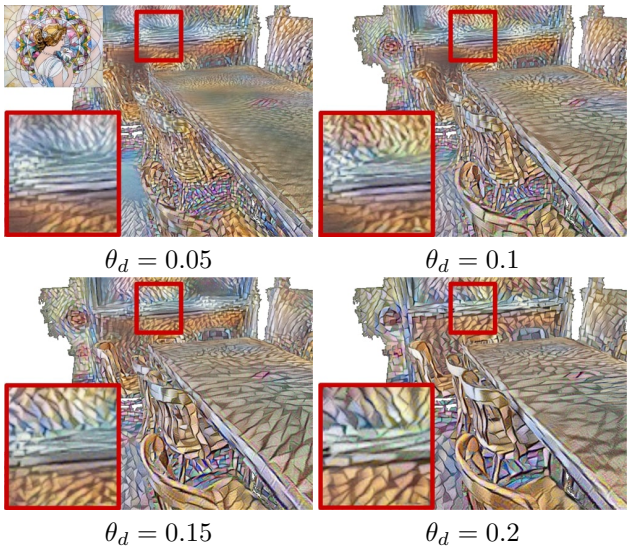
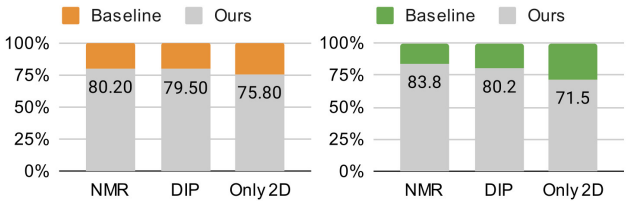


Figure 9. Variation of the minimum depth θ_d from 0.05 to 0.2 meters. Increasing its value leads to overall larger stylizations, while still retaining relative size differences within the scene. This can be used to fine-tune the stylization to a satisfying look.



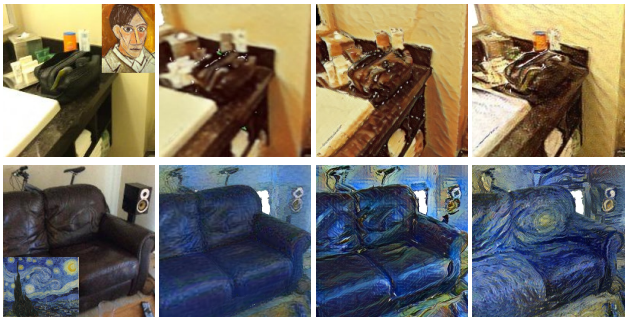
(a) Stretch (b) Size

Figure 10. We conduct a user study and ask subjects to select the results where stylization patterns (a) have less visible stretch and (b) are smaller in the background. The numbers indicate the percentage of preference for our method.

ages with a VST method and optimize an RGB texture from these images (VST→Tex). However, we might only have access to a sparse set of images in some scenarios. Due to inconsistencies between stylized frames (e.g., caused by illumination changes), the optimized texture is blurrier, as well. Our method is 3D-consistent by combining stylization and texture optimization over all available images directly.

4.4. Runtime Comparison

We propose an optimization-based NST method, that converges in roughly 3 hours on a single RTX 3090 GPU. After optimization, we can use the texture in traditional graphics pipelines and achieve real-time rendering, similar to [32, 42]. In contrast, model-based NST [25, 32, 35] might take days to train and needs a forward-pass at inference. However, these methods can generalize across scenes, whereas we need to optimize a separate texture per-scene.



RGB (a) VST→Tex (b) Tex→VST Ours

Figure 11. Video style transfer (VST) can be combined with RGB texture mapping to stylize an indoor scene. Either, we optimize a texture from the output of VST (a), or we render trajectories from an RGB texture and then apply VST (b). The first row uses VST of Wang et al. [54] and the second Deng et al. [15], respectively. In comparison, we produce sharper details and less noise.

4.5. Limitations

By design, our method is a per-scene/per-style NST algorithm, i.e., we optimize each explicit texture image separately. Recent work in implicit texture representations [45] could enable training generative models for our task. We do not disentangle lighting and albedo, i.e., view-dependent effects in camera images can be visible in the stylized texture. One could leverage neural rendering techniques to train a relightable stylization model [50]. Incomplete mesh reconstructions lead to holes in rendered poses, which can be reduced by employing mesh completion techniques first [11, 13, 14]. Similarly, an insufficient number of poses may lead to unobserved surfaces during optimization, i.e., we do not hallucinate texture. Inpainting techniques [50] could be utilized to complete those texels.

5. Conclusion

We have shown a method to stylize the mesh of room-scale indoor scene reconstructions. We lift style transfer to the 3D domain by optimizing a texture only through 2D images. Our method makes use of depth and surface normals of the mesh to achieve uniform world space stylization without view-dependent artifacts. For that, we split the loss calculation into image parts and stylize coarse and fine details separately. The explicit texture representation allows for real-time rendering of the scene after optimization.

Acknowledgements

This project is funded by a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant Scan2CAD (804724), and the German Research Foundation (DFG) Grant Making Machine Learning on Static and Dynamic 3D Data Practical. We also thank Angela Dai for the video voice-over.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2614–2623, 2019. 1

[2] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.*, 36(4):106–1, 2017. 1, 2

[3] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. Psnet: A style transfer network for point cloud stylization on geometry and color. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3337–3345, 2020. 2

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 4, 5, 6

[5] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1105–1114, 2017. 2

[6] Xinghao Chen, Yiman Zhang, Yunhe Wang, Han Shu, Chunjing Xu, and Chang Xu. Optical flow distillation: Towards efficient and stable video style transfer. In *European Conference on Computer Vision*, pages 614–630. Springer, 2020. 2

[7] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork, 2021. 1, 2

[8] Tai-Yin Chiu and Danna Gurari. Iterative feature transformation for fast and versatile universal style transfer. In *European Conference on Computer Vision*, pages 169–184. Springer, 2020. 2

[9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 4

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 4, 5, 6

[11] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2020. 8

[12] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)*, 2017. 1

[13] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018. 8

[14] Angela Dai, Yawar Siddiqui, Justus Thies, Julien Valentin, and Matthias Nießner. Spsg: Self-supervised photometric scene generation from rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1747–1756, 2021. 8

[15] Yingying Deng, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, and Changsheng Xu. Arbitrary video style transfer via multi-channel correlation. *arXiv preprint arXiv:2009.08003*, 2020. 2, 8

[16] Arnaud Dessein, William AP Smith, Richard C Wilson, and Edwin R Hancock. Seamless texture stitching on a 3d mesh by poisson blending in patches. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2031–2035. IEEE, 2014. 1, 2

[17] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, John F Hughes, and J Hughes. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996. 4, 5

[18] Chang Gao, Derun Gu, Fangjun Zhang, and Yizhou Yu. Reconet: Real-time coherent video style transfer network. In *Asian Conference on Computer Vision*, pages 637–653. Springer, 2018. 2

[19] Wei Gao, Yijun Li, Yihang Yin, and Ming-Hsuan Yang. Fast video multi-style transfer. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3222–3230, 2020. 2

[20] 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 (CVPR)*, June 2016. 1, 2, 3, 4

[21] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3985–3993, 2017. 2, 3, 4

[22] Agrim Gupta, Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Characterizing and improving stability in neural style transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 2

[23] Fangzhou Han, Shuquan Ye, Mingming He, Menglei Chai, and Jing Liao. Exemplar-based 3d portrait stylization. *arXiv preprint arXiv:2104.14559*, 2021. 2

[24] Filip Hauptfleisch, Ondřej Texler, Aneta Texler, Jaroslav Křivánek, and Daniel Sýkora. StyleProp: Real-time example-based stylization of 3d models. *Computer Graphics Forum*, 39(7):575–586, 2020. 2

[25] Hsin-Ping Huang, Hung-Yu Tseng, Saurabh Saini, Maneesh Singh, and Ming-Hsuan Yang. Learning to stylize novel views. *arXiv preprint arXiv:2105.13509*, 2021. 1, 2, 5, 6, 8

[26] Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 3dlite: towards commodity 3d scanning for content creation. *ACM Trans. Graph.*, 36(6):203–1, 2017. 1, 2

[27] Jingwei Huang, Justus Thies, Angela Dai, Abhijit Kundu, Chiyu Jiang, Leonidas J Guibas, Matthias Nießner, Thomas

Funkhouser, et al. Adversarial texture optimization from rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1559–1568, 2020. 1, 2

[28] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 2

[29] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 1, 2

[30] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 2

[31] Nikolai Kalischek, Jan D Wegner, and Konrad Schindler. In the light of feature distributions: moment matching for neural style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9382–9391, 2021. 2

[32] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018. 1, 2, 5, 6, 7, 8

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[34] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10051–10060, 2019. 2

[35] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, volume 40, 2021. 1, 2, 8

[36] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2479–2486, 2016. 2

[37] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3809–3817, 2019. 2

[38] Xiao-Chang Liu, Ming-Ming Cheng, Yu-Kun Lai, and Paul L Rosin. Depth-aware neural style transfer. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, pages 1–10, 2017. 3

[39] Xiao-Chang Liu, Xuan-Yi Li, Ming-Ming Cheng, and Peter Hall. Geometric style transfer. *arXiv preprint arXiv:2007.05471*, 2020. 2, 4

[40] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4905–4913, 2016. 4

[41] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 768–783, 2018. 2

[42] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 3(7):e12, 2018. 1, 2, 3, 5, 6, 7, 8

[43] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011. 1

[44] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013. 1

[45] 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. 8

[46] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German conference on pattern recognition*, pages 26–36. Springer, 2016. 2

[47] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126(11):1199–1219, 2018. 2

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[49] Daniel Sýkora, Ondřej Jamriška, Ondřej Texler, Jakub Fišer, Michal Lukáč, Jingwan Lu, and Eli Shechtman. StyleBlit: Fast example-based stylization with local guidance. *Computer Graphics Forum*, 38(2):83–91, 2019. 2

[50] Ayush Tewari, O Fried, J Thies, V Sitzmann, S Lombardi, Z Xu, T Simon, M Nießner, E Tretschk, L Liu, et al. Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, pages 1–320. 2021. 8

[51] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 3, 5

[52] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 2

[53] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *European conference on computer vision*, pages 836–850. Springer, 2014. 1, 2

[54] Wenjing Wang, Shuai Yang, Jizheng Xu, and Jiaying Liu. Consistent video style transfer via relaxation and regularization. *IEEE Transactions on Image Processing*, 29:9125–9139, 2020. 2, 8

[55] Xide Xia, Tianfan Xue, Wei-sheng Lai, Zheng Sun, Abby Chang, Brian Kulis, and Jiawen Chen. Real-time localized photorealistic video style transfer. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1089–1098, 2021. 2

[56] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12456–12465, 2021. 2, 3

[57] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014. 1, 2

[58] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997. 5