Eurographics Symposium on Rendering (DL-only Track) (2021)
A. Bousseau and M. McGuire (Editors)

MatMorpher: A Morphing Operator for SVBRDFs

Alban Gauthier[1], Jean-Marc Thiery[1], Tamy Boubekeur[2]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[2]Adobe

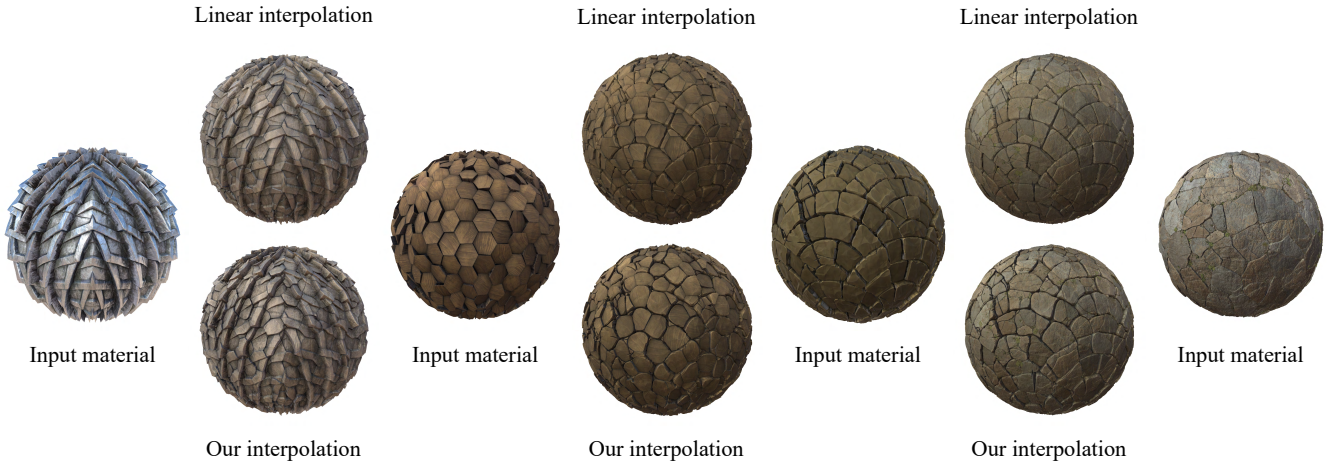

Figure 1: *Our operator provides a structure-preserving morphing between pairs of materials. It prevents the washed-out effect caused by linear blending and enhances the shape features during the interpolation between structured, semi-structured and unstructured materials.*

Abstract

We present a novel morphing operator for spatially-varying bidirectional reflectance distribution functions. Our operator takes as input digital materials modeled using a set of 2D texture maps which control the typical parameters of a standard BRDF model. It also takes an interpolation map, defined over the same texture domain, which modulates the interpolation at each texel of the material. Our algorithm is based on a transport mechanism which continuously transforms the individual source maps into their destination counterparts in a feature-sensitive manner. The underlying non-rigid deformation is computed using an energy minimization over a transport grid and accounts for the user-selected dominant features present in the input materials. During this process, we carefully preserve details by mixing the material channels using a histogram-aware color blending combined with a normal reorientation. As a result, our method allows to explore large regions of the space of possible materials using exemplars as anchors and our interpolation scheme as a navigation mean. We also give details about our real time implementation, designed to map faithfully to the standard physically-based rendering workflow and letting users rule interactively the morphing process.

CCS Concepts

• ***Computing methodologies*** → *Texturing; Reflectance modeling;*

1. Introduction

The physically-based rendering (PBR) workflow [Bur12] [Kar13] has become a standard for video games, visual special effects, product design and architecture, enabling developers and artists to create and share ready-to-use photorealistic materials among a wide variety of applications. In this workflow, 3D surfaces are mapped to a 2D texture space where their Spatially Varying Bidirectional Reflectance Distribution Functions (SVBRDF) are encoded as a set of bitmap images called PBR maps. These maps (e.g. Albedo, Normal, Roughness, ...) represent interpretable physically-based quantities while modeling good and versatile approximations of real world materials for the creative industries. The maps can be either

reconstructed from real-world photographs or generated procedurally, using specialized tools such as *Substance Designer* or *Quixel Mixer*. Unfortunately, both of these approaches to PBR material authoring require advanced skills and a significant amount of time to model convincing materials to be used by photorealistic renderers.

In this paper, we propose a new structure-preserving material interpolation operator which allows to create new materials by simply blending two existing ones while preserving their dominant features all along the interpolation. We designed this operator to be: **(i)** fast enough to provide real-time feedback, **(ii)** easily controlled using an intuitive blending map, **(iii)** feature-preserving across the multiple material channels. We stress the need of our technique to support interactive editing of high resolution SVBRDF maps up to a resolution of 4K.

Our method is based on a transport mechanism which continuously transforms the individual input PBR maps into their destination counterparts in a feature-sensitive manner. Exploring such high dimensional space could be naively achieved through linear interpolation but would then suffer from unwanted overlap of structures, ghosting artifacts and a lack of sharpness in the output. To alleviate these problems, we take inspiration from the image morphing community [Wol99] and first detect salient edges in the materials and create a mapping in the form of a transport grid in order to preserve important visual features during the interpolation. Each PBR map is then carefully interpolated to avoid detail losses and ghosting stemming from the naive interpolation. More precisely, we propose the following contributions:

- an efficient operator to explore and design-by-interpolation novel SVBRDFs based on an existing collection of materials,
- a transport grid model to guide the morphing process from the dominant mesostructures of the materials,
- novel real-time detail-preserving mechanisms to blend albedo and normal vectors consistently in this context.

2. Related work

Exemplar texture synthesis [WLKT09] [BZ17] provide means to generate numerous variations of a texture, starting from an exemplar pattern. Specific variations of input textures have been studied, such as weathering texture simulations [BKCO16] which can be used to synthesize general tileable, inhomogeneous and directional textures [MJH*17] [ZSL*17], but such techniques act on a single input RGB texture. [RLW*09] produce a 3D texture tile from two input RGB textures and their feature masks to allow for spatially varying texturing of surfaces. We choose a more lightweight approach, as their technique requires both high storage and long precomputation for high resolution textures. Recently, Guel et al. [GAD*20] presented a semi-procedural approach that avoids drawbacks of procedural textures and leverages advantages of data-driven methods. This approach allows for material structure interpolation but requires a binary segmentation of the input material, does not preserve the input features and is spatially but not temporally stable.

The work of Matusik et al. [MZD05] is closely related to ours. They present a system for designing novel textures starting from an input database, allowing to smoothly interpolate textures in a purposefully created space. Their work applies to RGB textures but not SVBRDF maps, which lie in incompatible spaces (color, spatial components, and distribution coefficients) and do not exhibit a complete spatial redundancy. We provide a thorough comparison to their method, which we reimplemented. [KPRN11] also focus on RGB texture interpolation, by optimizing for the complete texture metamorphosis, under an optical flow framework. In contrast, we optimize for an advection field retargeting salient structures, before blending retargeted signals in a secondary phase (similar to [MZD05]). Schuster et al. [STSK20] extended patch-based texture generation to meshes, which uses a multiscale optimization to minimize the visual artifacts between patches. They propose height-based blending as well as a histogram preserving blending (similar to [HN18] [Bur19]) as a mean of interpolating between SVBRDF maps. These methods do not tackle the problem of blending between two distinct materials however.

Two successive works [HN18] [Bur19] studied the synthesis of infinite textures using a randomized texture tiling of a stochastic input, preserving sharpness along the boundaries of each tile. As we observed that the approach to RGB texture blending presented in [MZD05] falls short with high-resolution images, we take inspiration from the aforementioned methods to preserve the albedo sharpness in our morphing operator.

Material editing methods allow for the direct manipulation of existing BRDFs or SVBRDFs (see [SPN*16] for a comprehensive review of material design and editing methods). [WTL*06] and [WCPW*08] navigate through the space of appearance by modeling the time-varying surface response or manipulating homogeneous diffuse BSSRDFs. [DRCP14] propose *editing spaces* for material parameters, providing common editing operations such as scaling, curve fitting and interpolation. Our method shares with this work the local frames editing approach. However our method differs in its ability to morph the mesostructures from one SVBRDF to another to explore the space of possible appearances.

Image Morphing and Manipulation. *Image Retargeting* [SS09], [RGSS10] address natural images and videos scaling/cropping, including seam-carving [AS07], scale-and-stretch [WTSL08] and distortion-free shape deformation [KFG09]. [LWX*09] focus on texture-guided refinement, by leveraging the texton appearance from one input texture and the distribution of the other. These techniques apply to RGB images only and do not provide interpolation between two input images, but rather focus on the deformation.

Patch-based methods. Fang et al. [FH07] build upon patch-based texture synthesis to deform shapes while avoiding stretch and compression. Barnes et al. [BSFG09] provide an interactive editing tool based on a quick randomized algorithm for finding approximate nearest neighbor matches between image patches, with applications in retargeting and reshuffling. Darabi et al. [DSB*12] proposed to synthesize a transition region between two source images leveraging an enriched patch-based search, a screened Poisson equation solver and a sharpness preserving norm. Mechrez et al. [MSZM19] base their optimization framework on a patch-based manipulation to enhance natural images.

[RSK10] is related to our work and proposes a patch-based method to create spatial interpolations between two RGB textures but differs from ours in several ways. This method is not fit for interactive design and requires a user-supplied distribution mask prior to the patch interpolation. In contrast, our method can leverage a blending mask interactively. In their work, manually crafted feature maps are used – instead of the compass operator [MZD05], which is deemed unsatisfactory by [RSK10]. Last but not least, their method is fundamentally a spatial interpolation method between non-superposed inputs (an *in-painting* method) whereas ours is a temporal interpolation method between superposed inputs. Image Melding [DSB*12] produces similar results than [RSK10] with a more lightweight optimization.

Contour detection. Contours are known to play a major role in human visual perception [PP11]; the cues they provide inspired a number of methods coming from the vast image processing literature. However, contour detection remains an ill-posed problem, somewhat dual to the image segmentation problem. Recently, Convolutional Neural Networks have proven their efficiency at addressing such problems, detecting contours of foreground objects as opposed to regions in the background [MPTAVG17]. These methods are trained on natural images, which differ significantly in structure from PBR maps, for which the notions of foreground/background are not well defined. They are also costly to evaluate, making them not ideal for interactive manipulation. The resulting contours appear thicker and blurrier than classical approaches based on local operators, which we adopt in our framework. In particular, Grompone Von Gioi et al. [GvGR16] proposed an Unsupervised Smooth Contour Detection. They present an efficient algorithm producing sub-pixel contour detection, without the need to learn a distribution prior to the detection.

Optimal Transport provides a framework for smooth interpolation and has been used for images [MÍ11] [XFPA14], BRDFs [BVDPPH11], 3D shapes [Lév15, SDGP*15], point clouds [BRPP15, BC19], color histograms [BPC16] and textures [RPDB12]. Recently, Nader et al. [NG19] proposed to quickly compute continuous transport maps between 2D probability densities discretized on uniform grids. Unfortunately, this framework does not aim at preserving structures during interpolation, but rather minimizing the cost of transforming one distribution to the other. Such costs are sensitive to outliers, in the sense that all mass is required to be transported to a target distribution.

Deep Neural Networks are often used to perform non-linear image transformation by leveraging massive image datasets. Navigating through the latent space stemming from these large datasets can indeed be interpreted as a form of advanced interpolation. For instance, Upchurch et al. [UGP*17] leverage a simple deep convolutional neural network and interpolate linearly among deep convolutional features, allowing high-level semantic transformations of human faces. Zhu et al. [ZPIE17] propose a Generative Adversarial Network (GAN) [GPAM*14] for learning to translate an image from a source to a target domain in the absence of paired examples, which was later improved and applied to a dataset of star faces [CCK*18]. Zhu et al. [ZKSE16] proposed to learn the natural image manifold directly from data using a GAN, allowing for image interpolation with user control over the object's shapes. Effland et al. [EKP*20] propose a multiscale feature space approach incorporating a deep convolutional neural network. Deep learning methods have recently proven their efficiency at synthesizing 2D and 3D textures [SCO17] [GRGH20] as well as interpolating between them [YBS*19] [HMR20]. [VDKCC20] study texture interpolation and synthesis with deep neural networks, providing insights about the distribution of CNN activations of natural textures. These techniques work well with stochastic textures but the models tend to overfit to the trained data, and always fail to preserve salient structures and autosimilarities in the inputs. MaterialGAN [GSH*20] requires an inverse rendering optimization to project the SVBRDF in the latent space. The interpolation of the latent variables produces temporally stable interpolation but fails to preserve both input textures. With respect to all these methods, and in the particular context of SVBRDFs, our method does not require lengthy training and only uses two input materials to produce a continuous, structure preserving interpolation, with no need to learn an a priori distribution. Indeed, our approach can feed the learning ones by providing an advanced mechanism to amplify the data set **before** training, as a substitute to the simple linear interpolations performed by e.g., Deschaintre et al. [DAD*18] or Guo et al. [GSH*20] to populate their training set.

3. Overview

Let us denote an input SVBRDF as a function

$$M\colon [0,1]^2 \to \{[0,1]^3, S^2, [0,1], [0,1], [0,1]\}$$

which associates to each point of the unit square domain $\Omega := [0,1]^2$ a diffuse albedo, a normal vector as well as roughness, metallicness and height (or displacement) scalar values. Starting from two input materials M_1 and M_2, our method (Figure 2) aims at producing a new SVBRDF M_α by morphing between the maps of M_1 and M_2 with a ratio α. To do so, we first compute a *transport grid* (Section 4) guided by the contours detected out of M_1 and M_2, and use it to morph each individual map in real time. Secondly, we use two novel operators (Section 5) to preserve the appearance of M_1 and M_2 during the morphing. More precisely, we use a *histogram aware color blending* to preserve the sharpness of the input albedo maps and introduce a normal-and-height cross-interpolation mechanism to recompute the normal from the interpolated geometry and preserve details. Propelled by our transport grid, these operators yield the final interpolated SVBRDF M_α. In its most general form, our morphing mechanism lets the user provide a spatially varying morphing ratio α given by a scalar field $\alpha_S\colon \Omega \to [0,1]$. This scalar field is defined over the same texture domain as M and its range values modulate a per-texel interpolation of M_α, performing a spatially-varying non-linear mix between M_1 and M_2.

4. Transport grid

Our transport grid models per-texel correspondences between M_1 and M_2. We design it specifically to avoid the default per-pixel correspondence of linear blending which yields unsatisfactory geometry during the interpolation. The transport grid aims at preserving

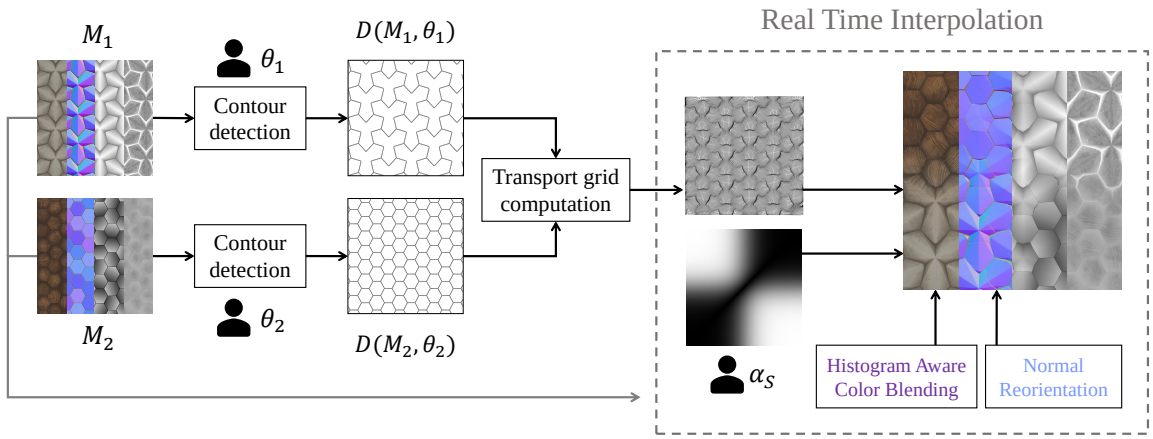

Figure 2: *Overview of our method.*

the structures present in the material as much as possible all along the interpolation. To do so, we first detect contours from the input materials before optimizing a deformation over the meshed unit square.

4.1. Contour detection

We seek feature lines present in the input materials to use them as a super structure driving the morphing process. Such lines typically separate materials regions where texel statistics are close to uniform. In our case, the different channels of each SVBRDF provide a rich space that we leverage to detect the dominant mesostructures. More precisely, we base our contour extraction on the parameter-free algorithm proposed by Grompone et al. [GvGR16] which produces locally consistent results across all SVBRDF maps in a matter of seconds.

By default, we run this algorithm on the heightmap of each material M. Since our method aims at preserving shape features during the interpolation, most heightmaps are sufficient to describe the geometry and produce meaningful contours. Optionally, the user can control the process by activating contour detection on a per-channel basis using a set of booleans $\theta = \{\theta_a, \theta_r, \theta_m, \theta_n, \theta_h\}$. This prevents unsatisfactory detection in materials where the heightmap fails to provide sharp features. The contour detection algorithm is fed with greyscale images. For the normal map, we simply retain its z component. The albedo map is converted to greyscale using the luminance channel. Secondly, we sample the resulting polylines in the selected maps before concatenating the resulting 2D point sets into a single contour sampling $\mathcal{C} = \{p_1, ..., p_k\}$ made of 2D points $p_i \in \Omega$. This yields a global contour extraction operator:

$$D(M, \theta) \to \mathcal{C}$$

that we apply on both M_1 and M_2.

4.2. Transport Map Generation

Our transport map provides a one-to-one correspondence for each texel between M_1 and M_2. We aim at transporting efficiently points $P = \{p_i \in \Omega\}_i$ sampled from the source material contours $\mathcal{C}_1$ in order to best align them onto a target point set $Q = \{q_j \in \Omega\}_j$ sampled from the target material contours $\mathcal{C}_2$.

Since we cannot make assumptions on the input such as, e.g., perfect sampling or one-to-one structure correspondence, we design our algorithm with the prime intent of being robust to outliers. To do so, we propose to minimize a transport energy modeling a kind of elastic sparse Iterative Closest Point method [BTP13]:

$$\mathbb{E}_{\text{fit}}(f) := \sum_{i,j} w_{ij} \|f(p_i) - q_j\|^s + \lambda \mathbb{E}_{\text{reg}}(f) \quad (1)$$
$$\text{s.t. } f(\partial\Omega) = \partial\Omega$$

w_{ij} being a localization kernel, $s \in]0, 2]$ ($s < 1$ resulting in increased robustness to outliers).

Iterative solver. Minimizing this energy is nontrivial for sparsity parameters $s < 2$. We present here an efficient iterative solver for this task. We start with a given configuration f (by default: Identity). In practice, at each iteration, we restrict the double summation by taking the k nearest neighbors of $f(p_i)$, and set the weights w_{ij} to 0 for any q_j further away from $f(p_i)$. Our implementation makes use of a Gaussian kernel with standard deviation 0.01. Further, we enforce harmonic (null Laplacian) deformations to regularize the solution.

We start with a uniform grid with vertices $\{v_{k,l} = (x_{k,l}, y_{k,l}) \in \Omega; (k,l) \in [0, N[^2\}$ stacked into the optimization unknowns vector $\mathbf{v}$, together with bilinear interpolation basis functions ($f(p_i) = A_i \cdot \mathbf{v}$, A_i being the row containing the bilinear coordinates of p_i in the input grid-mesh), and the standard four-point approximation of the Laplacian operator onto the grid-mesh ($\Delta f(v_{k,l}) := v_{k+1,l} + v_{k-1,l} + v_{k,l+1} + v_{k,l-1} - 4v_{k,l}$). In order to approximate the L^s norm, we use a standard reweighting scheme, and finally minimize at each iteration

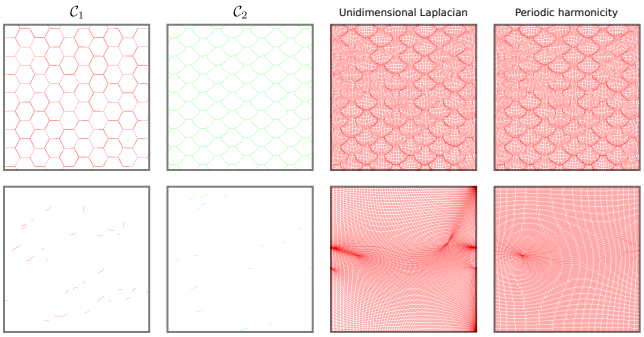

Figure 3: ***Boundary conditions analysis.*** *In the more general case, when dealing with dense structures to match (top), the boundary conditions impact the solution only locally. When dealing with structures to match that are very sparsely distributed (below), the boundary conditions strongly impact the solution globally, as most conditions are too loose to enforce strong regularization in these cases. Whether the inputs are tileable or not, we find that* periodic harmonicity *provides the best results in all cases.*

$$\mathbb{E}'_{\text{fit}}(\mathbf{v}) := \sum_{i,j\in NN(i)} w'_{ij}\|A_i\cdot\mathbf{v}-q_j\|^2 + \underbrace{\lambda\sum_{0<k,l<N-1}\|\Delta f(v_{k,l})\|^2}_{\text{interior regularity term}} + \underbrace{\mathbb{E}_\partial}_{\text{boundary terms}} \tag{2}$$

where $w'_{ij} = w_{ij}\max(\varepsilon, \|A_i\cdot\mathbf{v}^{\text{current}} - q_j\|)^{s-2}$ is used to approximate the L^s norm objective function (ε being a safety parameter here to avoid divisions by 0, $\varepsilon = 10^{-3}$ in our implementation).

In practice, we repeat this optimization T times using a Cholesky decomposition, and advect the solution with a step $\delta = t/T$ at iteration $t \in [1,T]$. This allows refining progressively the local matching $NN(i)$ (the k points in Q nearest from point $f^{\text{current}}(p_i) = A_i\cdot\mathbf{v}^{\text{current}}$), and typically results in better matchings when structures to be matched are not aligned properly or in presence of outliers.

Boundary conditions So far, we have not constrained the boundaries of the mesh explicitly. While typical boundary conditions such as Neumann or Dirichlet are used in a variety of transport problems in Computer Graphics, we advocate the use of ad-hoc boundary conditions that are better suited to our problem. We analyze in the following various boundary conditions and discuss their pros and cons.

Constraining *the boundary image* (i.e., $f(\partial\Omega) = \partial\Omega$) can be done trivially by removing the corresponding variables from the optimization, or almost equivalently, by adding a large penalty term onto those (e.g., $\mu|x_{0,l}-0|^2$, $\mu|x_{N-1,l}-1|^2$). The boundary term takes in this case the form:

$$\begin{aligned}\mathbb{E}_\partial += &\mu\sum_k |y_{k,0}-0|^2 + |y_{k,N-1}-1|^2 \\ &+\mu\sum_l |x_{0,l}-0|^2 + |x_{N-1,l}-1|^2\end{aligned} \tag{3}$$

Note that *tileability* (i.e., $f(\partial\Omega)\neq\partial\Omega$ and $\{f(\Omega)+i*(1,0)+j*(0,1); i,j\in\mathbb{Z}\}$ defines a partition of $\mathbb{R}^2$) can be enforced instead in a trivial manner similarly, e.g., by adding a large penalty such as $\mu|x_{N-1,l}-x_{0,l}-1|^2$.

Boundary *smoothness* can be enforced in several ways. We have studied unidimensional harmonicity as well as periodic harmonicity.

The former can be enforced by adding the following terms to $\mathbb{E}_\partial$:

$$\begin{aligned}\mathbb{E}_\partial += &\lambda\sum_{0<k<N-1}|2x_{k,0}-x_{k-1,0}-x_{k+1,0}|^2 \\ &+\lambda\sum_{0<l<N-1}|2y_{0,l}-y_{0,l-1}-y_{0,l+1}|^2\end{aligned} \tag{4}$$

while the latter is enforced by adding the following terms to $\mathbb{E}_\partial$:

$$\mathbb{E}_\partial += \lambda\sum_{(k,l)\in\partial[0,N-1]^2}\|\Delta f(v_{k,l})\|^2$$

while making sure that off-grid indices point to the correct variables inducing periodicity in the Laplacian operator (e.g. $x_{-1,l}$, resp. $x_{N,l}$, is replaced $x_{N-2,l}$, resp. $x_{1,l}$).

We have analyzed the impact of these conditions on various examples, and we found that *periodic harmonicity* results in increased stability in all cases, even if the input textures are not tileable (see Fig.3). Note that in that case, the output interpolation will not be constrained to be tileable, but will merely be constrained to be advected in a smooth manner around the boundaries.

5. Material Interpolation

With our transport grid in hand, we can now perform an interpolation that produces visually appealing motions of the input materials mesostructures. The second component of our method is dedicated to the blending applied to each texel during the interpolation. Our experiments revealed that linearly combining the input color (resp. normal) maps produces unsatisfactory results; therefore, we propose techniques to improve the blending in Section 5.1 (resp. 5.2) before describing our final SVBRDF morphing method (Section 5.3).

5.1. Histogram aware color blending

When interpolating colors linearly, details from each material tend to be lacking and colors appear dull. Matusik et al. [MZD05] who link this behavior to the averaging effect led by the blending suggest extracting high-frequency statistics and enforce these statistics on the interpolated result. This is achieved by capturing high-frequency content using a multiscale frequency decomposition. They use color histogram matching and enforce the histogram of the linearly blended texture decomposition to be an interpolation of the histograms of the input decompositions. We tested their

approach on albedo maps, and it appears the effect of the decomposition vanishes when working with high resolution textures (1K and 4K). Their interpolation focuses on 128x128 textures, which do not require a multiscale approach. A simpler way of interpolating texture is done by using histogram matching as in [MZD05], but with no prior decomposition of the images.

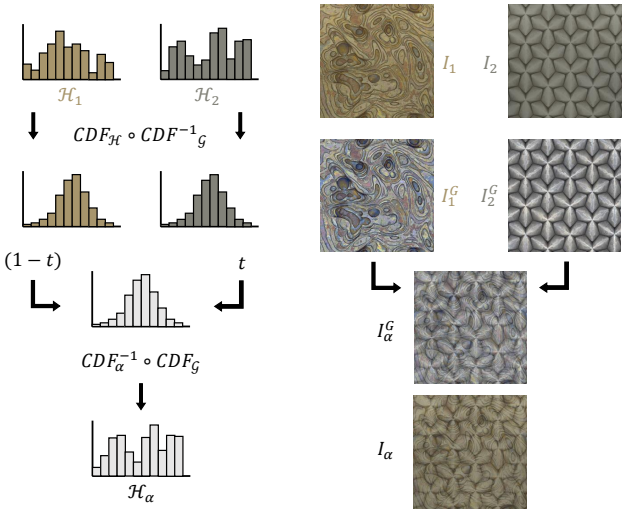

Figure 4: *Starting with albedos I_1 and I_2, we compute the gaussianized albedos I_1^G and I_2^G, interpolate them to get I_α^G and apply the inverse histogram composition which results in I_α. Our technique better preserves color sharpness compared to a linear interpolation.*

We take insights from Heitz et al. [HN18] who show that blending linearly between two random variables acts as a histogram convolution, resulting in a low variance in the histogram of blended textures. Even though this result is given under the assumptions that the texture's pixels are independent and identically distributed – which breaks in our case – the idea inspired our histogram aware blending operator (see Algorithm 1). We use both the Cumulative Distribution Function of each input, and the truncated gaussian distribution from Burley et al. [Bur19]. Similarly to Burley, we use gaussianization with a Soft-clipping Contrast Operator $S^*_{[\hat{G}]}$ to obtain more contrasted interpolation. We gaussianize independently each input and obtain I_1^G and I_2^G, which we then interpolate linearly before reverting the histogram equalization by applying successively the truncated gaussian CDF (CDF_G in Algorithm 1), and the interpolated inverse CDF of the input textures through a lookup table (LUT_α^{-1} in Algorithm 1). The pipeline is illustrated in Figure 4. The whole pipeline is executed per channel. In practice, the user can choose to run our histogram aware interpolation in either RBG or YCbCr color space to prevent a slight shift in hue which occurs with one or the other depending on the input material pair.

We compare the results of different linearly interpolated texture enhancements (sharpness preservation, histogram matching and gaussianization) in Fig. 13 and in the supplemental materials.

Algorithm 1: Histogram aware blending of texels p1, p2

$$p_\alpha \leftarrow (1-\alpha)\, I_1^G[(x,y)_{p1}] + \alpha\, I_2^G[(x,y)_{p2}]$$
$$W \leftarrow \sqrt{(1-\alpha)^2+\alpha^2}$$
$$p_\alpha \leftarrow S^*_{[\hat{G}]}(p_\alpha, W)$$
$$p_\alpha \leftarrow LUT_\alpha^{-1}[CDF_G(p_\alpha)]$$

with $LUT_\alpha^{-1} = (1-\alpha)\, CDF_1^{-1} + \alpha\, CDF_2^{-1}$

5.2. Height and Normal cross-interpolation

5.2.1. Normal map preprocessing

For procedural PBR materials, the normal and height maps are often created following these steps: first, artists create displacement information which corresponds to the base geometry. This information is stored in a greyscale height map. Second, they add shading details – *bump mapping* – on top of the base geometry and store the resulting normal in pre-displacement tangent space, instead of the local space induced by the displacement of the geometry (post-displacement tangent space). Fetching normals from pre-displacement tangent space simplifies the computations at runtime and avoids the regeneration of the post-displacement tangent space, which does not vary across time in classical scenarios. At runtime, the per-vertex tangent-space basis is provided and helps the reorientation of the normal according to the geometry. However, this results in wrongly oriented normals when modifying the displacement factor during rendering if the tangent frame is not recomputed (Fig. 5, bottom).

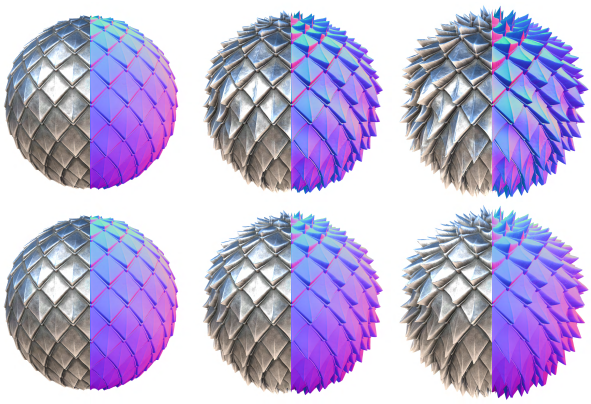

Figure 5: ***Normal map and geometric normal mismatch****. At the top, the normals (right hemisphere) are computed from the displaced surface, at the bottom, a simple texel fetch in the normal map followed by a change of frame using the TBN matrix, overlayed on the displaced geometry. Displacing the geometry with a varying height factor creates a "painting effect" (left hemisphere), where the normal reflections seem to be painted on the surface, no matter the orientation of the tiles.*

In our context, the interpolation between structures is non-trivial and such an approach fails to produce a valid interpolated normal. Besides, the extra details lie in post-displacement tangent

space, and must consequently be blended in this basis. Similarly to [DRCP14], we use a decomposition of the normal map w.r.t. the geometric normal. Specifically, we start by *reverse-engineering* the normal baking process, in order to access these post-displacement details (bump maps). At runtime, the geometry i.e., the height map information, is linearly interpolated. The details are combined using a spherical linear interpolation, and the resulting detail normals are consistently added back to the geometry.

To further support our argumentation about invalid normals interpolation, we illustrate the need for a joint interpolation of the height and the normal maps in Fig. 6. When interpolating linearly the geometry between the flat solid shape and the dotted shape (Fig. 6, left), one can see that the slope at a point located between the two plateaus follow the relation $\theta(\alpha) = \arctan(\alpha h/d)$ (the blue curve), and not simply the one given by the SLERP interpolation of the normal map (i.e., $\alpha\arctan(h/d)$, the red curve). One has therefore to distinguish between the *macro-structure* orientation given by the height map and the *meso-structure* orientation, which is described by the deviation of the normal from the macro-structure orientation. We base our height-and-normal interpolation operator on this observation. In order to preserve sharp geometric features during the interpolation, we perform a two-scale analysis of the normal map by decomposing it into a coarse and a detail map, recomposed after interpolation.

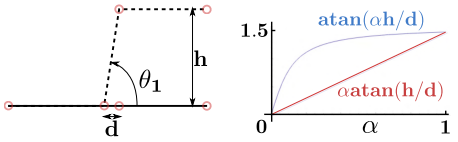

Figure 6: *The linear interpolation of the height map is inconsistent with the spherical linear interpolation (SLERP) of the normal map. The plots were computed for $h/d = 10$.*

5.2.2. Normal reorientation pipeline

Following [DRCP14] and Mikkelsen [Mik19] layered approach, we propose a normal map decomposition summarized in Fig. 7. Basically, we first compute a detail map for each material, using the regular normal map and a coarse normal map derived from the height map. Then we interpolate linearly the heightmaps, and compute a coarse normal map from the geometry. Lastly, we use a spherical linear interpolation on the detail maps, extract the proper normal from the geometry, and add the details to get the final interpolated normal map.

Algorithm 2: Normal Reorientation algorithm

$\mathbf{n}_1^h \leftarrow normalize((\nabla_x H_1, \nabla_y H_1, h_1))$
$\mathbf{n}_2^h \leftarrow normalize((\nabla_x H_2, \nabla_y H_2, h_2))$
$\mathbf{d}_1 \leftarrow getDetails(\mathbf{n}_1^h, \mathbf{n}_1)$
$\mathbf{d}_2 \leftarrow getDetails(\mathbf{n}_2^h, \mathbf{n}_2)$
$\mathbf{d}_\alpha \leftarrow slerp(\mathbf{d}_1, \mathbf{d}_2, \alpha)$
$\mathbf{n}_\alpha^h \leftarrow normalize(cross(dFdx(position), dFdy(position)))$
$\mathbf{n}_\alpha \leftarrow addDetails(\mathbf{n}_\alpha^h, \mathbf{d}_\alpha)$

To extract a detail map vector $\mathbf{d}$ w.r.t. the original normal and

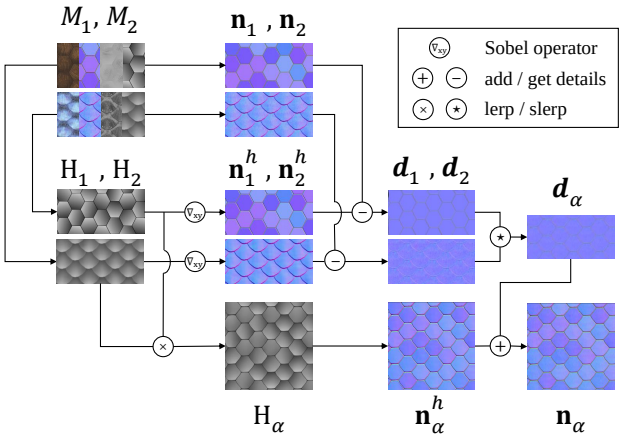

Figure 7: *Normal reorientation pipeline.*

height map (method *getDetails* in Alg. 2) (i) we define the coarse base normal $\mathbf{n}^h$ from the height map, (ii) we build a transform R that rotates $\mathbf{n}^h$ into the normal map vector $\mathbf{n}$ containing a higher frequency signal and (iii) we apply R to $\mathbf{z}$, the unit normal of the plane (see Fig. 8). We then combine these details by using a spherical linear interpolation, and we add them to the interpolated coarse normal similarly to Barré-Brisebois and Hill [BBH12].

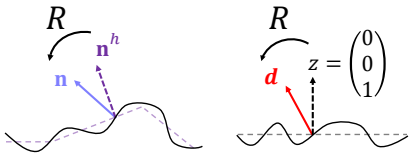

Figure 8: ***Normal reorientation***. *We compute the rotation R from the coarse geometry $\mathbf{n}^h$ to the detailed one $\mathbf{n}$ and apply it to a unit vector perpendicular to the plane $\mathbf{z}$ to obtain the detail vector $\mathbf{d}$.*

To compute a normal map from each material's height map, we use a Sobel filter which provides smoother results than a simple gradient [WBW20]. We use 16 bits height maps for better precision. However, recovering normals from the height maps with the same orientation and amplitude as the original normal maps requires the knowledge of the global height factor h used to compute it originally. Therefore, we retrieve this factor as follows: let $H\colon [0,1]\times[0,1] \to [0,1]$ be the height map and ∇_x, ∇_y the vertical, resp. horizontal Sobel operators, such that:

$$\nabla_x H = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} * H \quad \text{and} \quad \nabla_y H = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * H$$

To retrieve h, we solve the following optimization problem, where N is the original normal map and N^h the normal map recovered from the heightmap:

$$\underset{h\in[0,1]}{\operatorname{argmin}} f(h) = \textstyle\sum_{(i,j)} d(n^h_{i,j}, n_{i,j}) \quad \text{with} \quad d(x,y) = \|acos(x \cdot y)\|_1$$

In practice we compute 128 normal maps from the height map in a compute shader with a varying height factor $h \in \{i/127, i \in$

$[0,127]\}$ prior to the rendering. Experimentally we found out that all the computed f functions show a convex behavior, thus allowing to easily recover the h parameter.

5.3. Practical Implementation

Our transport map f provides a one-to-one correspondence between both materials. As previously stated, the user can provide a scalar field α_S with values between 0 and 1 at each texel to control the interpolation. This field is parameterized both spatially and temporally and drives the transport function:

$$f_S : \Omega \to \Omega, f_S(p) = (1-\alpha_S(p))p + \alpha_S(p)f(p)$$

For a point $p_1 \in \Omega$ describing a location in M_1 and the corresponding interpolation parameter α fetched from the α_S, we compute the value at the interpolated point p_α by linearly interpolating between its source p_1 and its target $f(p_1) = p_2 \in \Omega$, where we gather the corresponding SVBRDF parameters in M_2:

$$p_\alpha = (1-\alpha)p_1 + \alpha p_2, \tag{5}$$

At p_α, we map values of M_1 at p_1, and those of M_2 at p_2. For the height and metallic maps, we linearly blend the values to obtain those of the final maps - note that PBR materials being physically inspired rather than physically correct [Bur12], linearly interpolating the metallic term is considered the industry de facto standard for interpolation. We observe that the transition from shiny metal to a rusty or worn aspect gives a pleasing result. Regarding roughness, [DRCP14] proposes to interpolate the cumulated distribution functions of the BRDFs, which can be seen as a form of optimal-transport-based interpolation. While this framework provides satisfying results and is mathematically well-grounded, we chose to simply interpolate the square mapping of the roughness, as advocated in Disney's principled BRDF model [Bur12], as artists observe empirically that this results in a perceptual linear change of the materials (see Section 5.4: Specular D details). For the albedo and normal, we use our previously introduced operators. Those values are fed to a standard microfacet model with a GGX normal distribution function for rendering.

In practice, we run this process in real time using the programmable rendering pipeline of modern GPUs. We first rasterize the displaced grid using rest positions as a color attribute. The displacement of each vertex v is computed as $f_S(v)$ (i.e., using the α values at their original position in α_S, see Equation 5), thus resulting effectively in a coarse linear rendering of $f_S^{-1} : \Omega \mapsto \Omega$. At each output texel q of the computed image, we first read the coordinate $p_1 = f_S^{-1}(q)$ of the texel m_1 from M_1 to blend, we then fetch α at position p_1, and we compute the coordinate $p_2 = f(p_1)$ of the texel p_2 from M_2 to blend. With these at hand, we then interpolate the materials' texels m_1 and m_2 accordingly, and finally shade the output fragment. Being real time, our method allows the user to vary the blending field α_S interactively using a brush metaphor or the parameters of a procedural texture.

6. Evaluation and Applications

In Fig. 11, we present a collection of interpolation results obtained with our method on a set of pairs of SVBRDFs. Note that videos

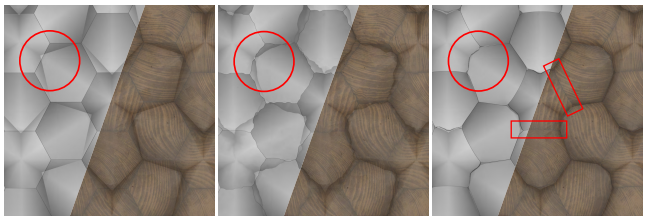

Figure 9: ***Transport grid ablation***. *Without our transport grid, i.e. using linear blending (left), undesired overlapping of structures along the edges tend to appear. [MZD05] technique (middle) fails to correctly register contours, due to imprecise feature maps. The resulting warpgrid exhibits discontinuities along borders due to the independent per-vertex optimization. Using our transport grid (right) allows for contour registration (circle) and creates a natural morphing even if a contour does not find a correspondent in the other material (rectangles).*

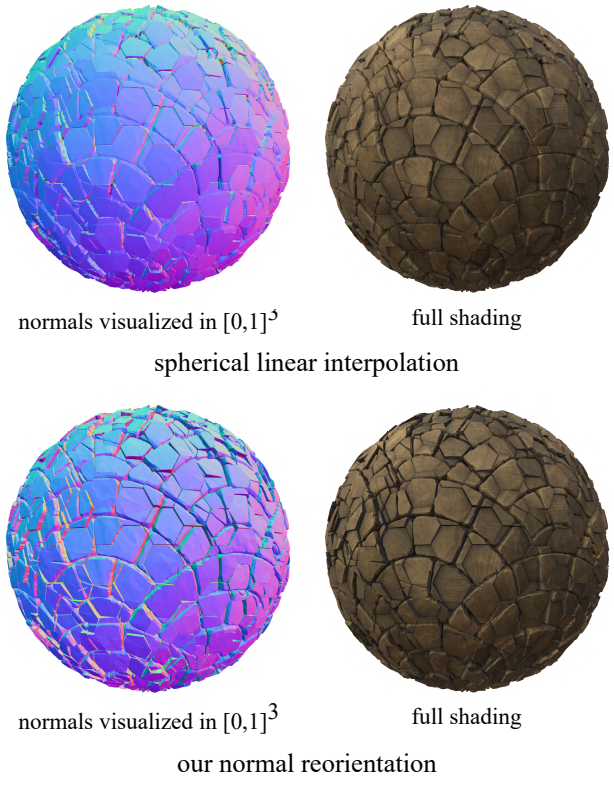

Figure 10: ***Normal reorientation ablation***. *We show a halfway interpolation between two materials. Our reorientation component better conveys the shape features all along the interpolation by computing a geometric normal with added details. No transport map was used for this example.*

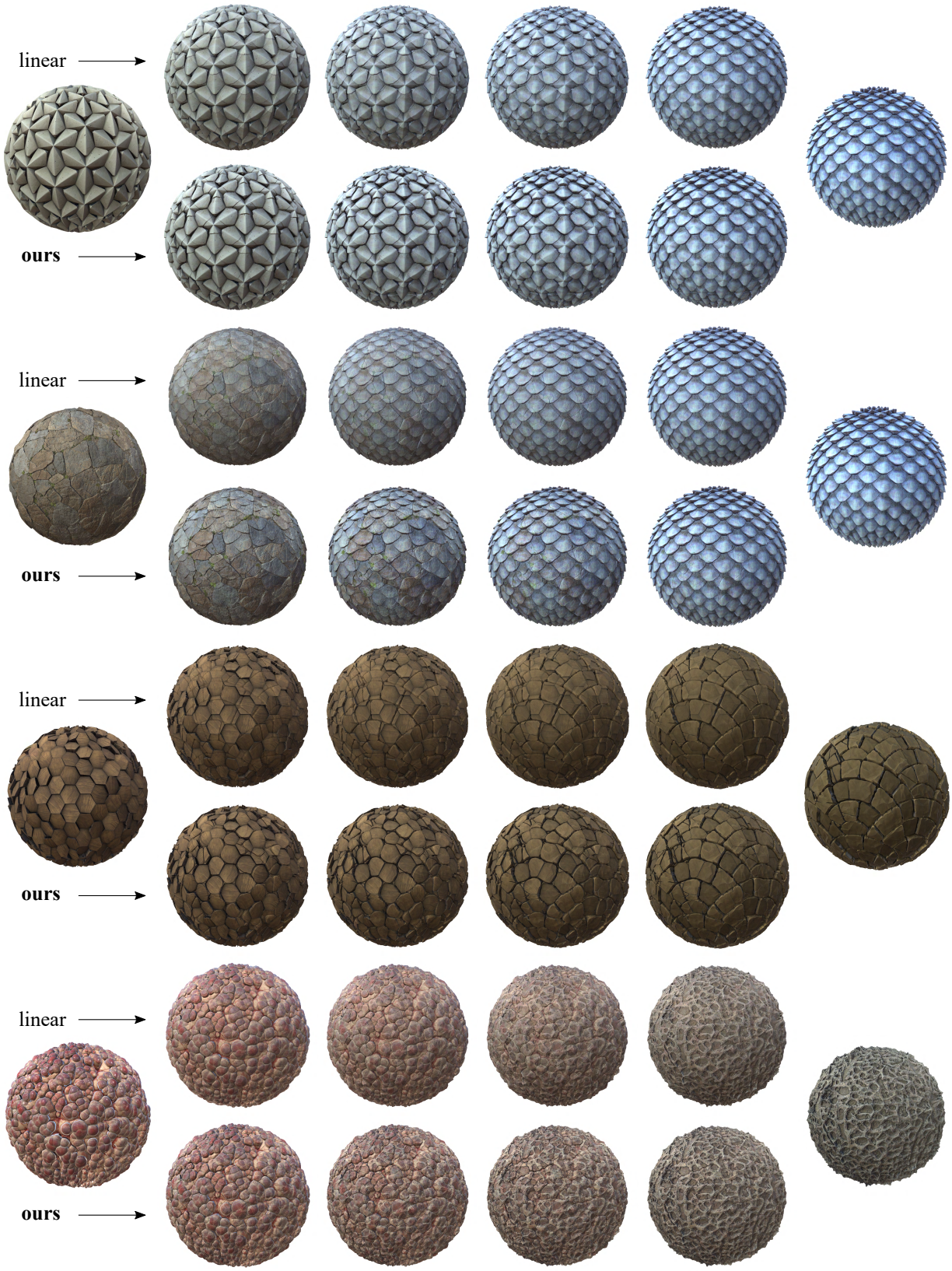

Figure 11: *Interpolated materials M_α with a linear blending (top row) compared to our technique (bottom row), using a uniform value for $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We stress that temporal interpolations are better viewed in videos (provided as supplemental materials).*

are better suited to visualize temporal interpolation, and that we provide video comparisons in the supplemental materials.

6.1. Ablation study

We evaluate the benefit of individual components of our approach in an ablation study.

Transport Grid. Our transport grid allows for a clean contour registration which morphs the structures of the first input material into the others. We show in Fig. 9 that when a contour from one material has no match in the other material, a ridge collapses or appears while preserving its shape. This is the desired transport behavior when outliers occur. We compare our transport grid with [MZD05], which fails to produce precise contour registration between maps.

Normal Reorientation allows to preserve the shape features of the geometry along the interpolation. Compared to spherical linear interpolation, which produces wrongly oriented normals, this component ensures proper detail preservation (see Fig. 10).

6.2. Comparisons

We reimplemented the method of [MZD05] in order to compare their results to ours, in the restricted context of RGB texture interpolation. We provide a comparison in Fig. 12 and in the supplemental materials, and explain the shortcomings of the method when dealing with high resolution SVBRDF maps. Their technique is composed of two main steps : the warp grid computation and the RGB texture interpolation. First, the warp grid computation reveals discontinuity artifacts, which result from the independent per-vertex optimization at each iteration. The method fails to provide a globally smooth warpgrid, and as a result creates discontinuities at the borders of each region inside materials. We also noted that processing a material as a 9D stack of greyscale images fails to create salient feature maps to blend. Therefore, we exhaustively selected the best feature maps combination for each of our example materials, visualizing the resulting halfway morphing between albedo maps. This results in warpgrids which are not able to consistently align features in materials. Second, the steerable pyramid decomposition followed by histogram matching falls short when dealing with high resolution textures. We applied the technique on both 128x128 textures (as in their work) and with 1K and 4K textures, and as the resolution increases, the effect of matching histograms on the filtered images fades progressively.

We noticed that in most cases, simply applying histogram matching [MZD05] to the whole image with no predecomposition yields better contrasted results. In Fig. 13, we provide a comparison between linearly interpolated textures. The advantage of this method over simpler histogram matching is that the histogram of the interpolated texture is known during interpolation, and does not require a pass through the whole image to compute the histogram of the interpolated texture. As explained previously, simpler histogram matching can be used as a way of sharpening the interpolated texture. Unfortunately, this method has a runtime complexity of $O(N)$ (N being the total number of pixels of each image) compared to the constant $O(1)$ complexity of the gaussianization (in that case the histogram matching happens during precomputation), which makes real-time editing possible.

	Timings	
	Contour Detection	Transport Grid
4K downsampled to 1K	3.1 ± 2.1 s	18.6 ± 6.7 s
	Gaussianization	Normal Compute
Ours 4K	350 ms	400 ms
	Single light	3 lights + IBL
Linear 4K	7.87 ± 0.07 ms	9.89 ± 0.09 ms
Ours 4K	15.10 ± 0.06 ms	19.41 ± 0.13 ms

Table 1: ***Timings***. *We render an interpolation between two materials for a total of about 50 million triangles under a single point light, and under three point lights and Image Based Lighting for the experiment of the last two lines of this table.*

We also compare our technique to MaterialGAN [GSH*20], which proposes the exploration of a learned distribution of material appearance by interpolation. To do so, we project input renderings in the latent space to recover noise and style vectors, that we use to create the final interpolation. The results are shown in the supplementary materials. MaterialGAN provides a temporally stable interpolation, but offers no control over the appearance, and fails to preserve the appearance of the input textures.

6.3. Performance

The transport grid precomputation depends on the number of contour samples detected in each material. We report the timings of the contour detection for a single map at 4K resolution (which is downsampled at 1K beforehands only for the contour detection phase), along with the transport grid computation in Table 1. For this experiment, all contour points were provided to the later stage (ranging from 10'000 up to 80'000) but a resampling can be done to achieve better performance. At runtime, the dynamic interpolation cost of our method is roughly twice higher than linear interpolation, still allowing for real time design of the interpolation scalar field α_S. In Table 1 we compare our method against a simple linear interpolation and report the precomputation timing, all measured on a Ryzen 5 5600X CPU running at 4.6GHz equipped with a Nvidia RTX 2080 Ti GPU.

7. Limitations and Future Work

Our method struggles with materials which contain only stochastic structures such as moss or dirt, and is favored when materials exhibit medium to large salient structures. We also assume a rough initial alignment of the two inputs when they are highly periodic, which could be automated but raises the question of preserving tileability if present. Similarly, the transport grid provides good contour matches when the mesostructures of the two inputs are spatially close. Otherwise, the interpolation results in a simple blend for the regions where no structures can be blended with. Last, even if most of our material examples present a low frequency periodicity in their structures, our technique does not exploit this selfsimilarity, which is an interesting direction for future work.

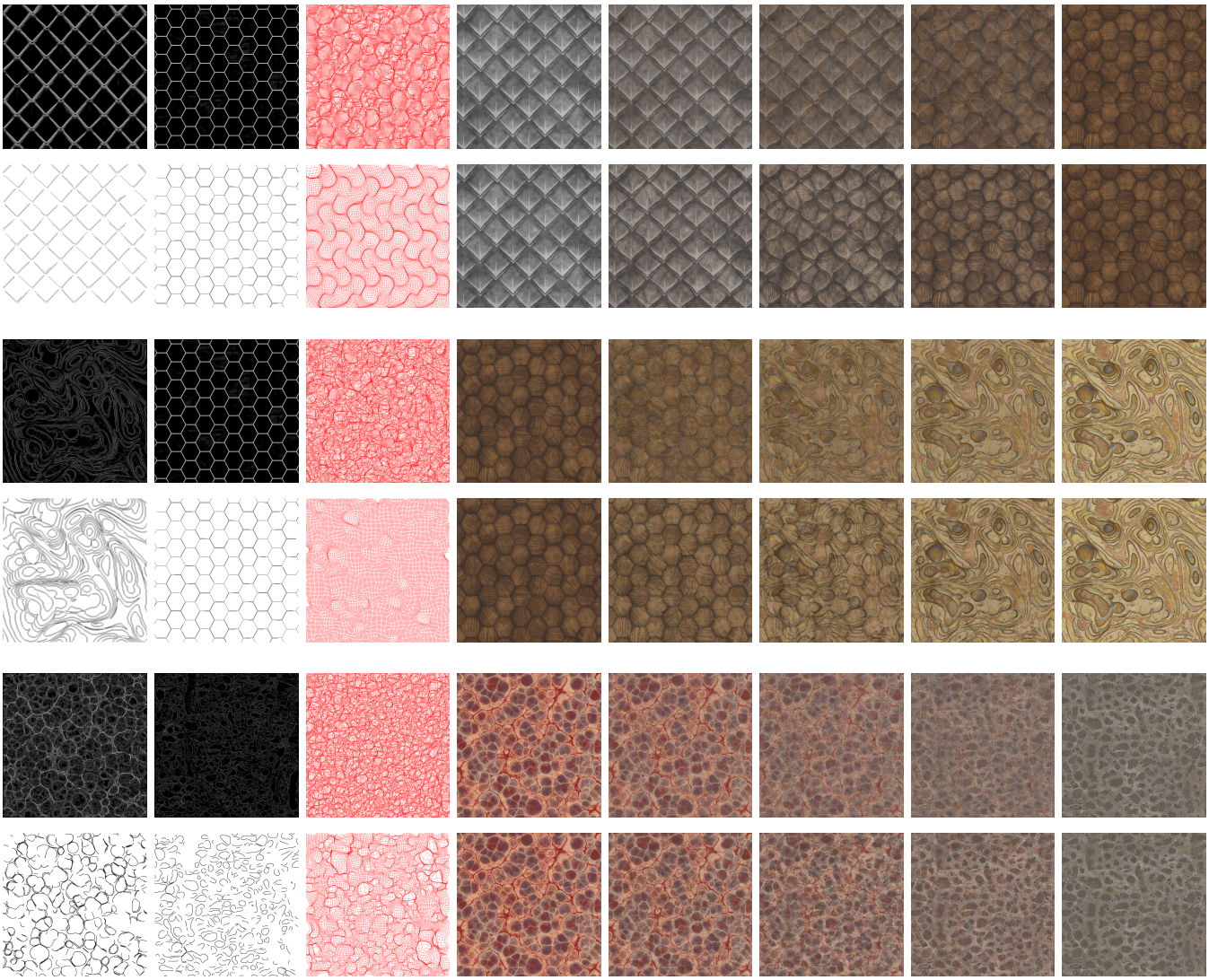

Figure 12: ***RGB texture interpolation*** *comparison between [MZD05] (top) and our method (bottom). For each example, we show the features used to compute the transport grid (visualized in red) and five interpolation steps from 0.0 to 1.0.*

8. Conclusion

We presented a novel structure-preserving operator for interpolating two SVBRDFs. To do so, we introduced a transport grid which guides the interpolation based on the dominant mesostructures detected in the input, and combined it with normal reorientation and histogram-aware color blending to better preserve details and sharp features. Our operator is fast enough to run in real time, providing a simple mechanism to explore material variations out of a set of exemplars, with a visual quality which is superior to previous methods, for a cost which is close to linear interpolation. Our approach proposes multiple techniques which can be used independently. For instance, the height factor retrieval from Section 5.2 can benefit workflows which produce both a geometry and its corresponding normal map. Last, our operator shall find application in material database amplification, as leveraged by deep nets in neural rendering.

References

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 papers*. 2007, pp. 10–es. 2

[BBH12] BARRÉ-BRISEBOIS C., HILL S.: Blending in detail. https://blog.selfshadow.com/publications/blending-in-detail/, 2012. 7

[BC19] BONNEEL N., COEURJOLLY D.: SPOT: sliced partial optimal transport. *ACM Transactions on Graphics 38*, 4 (2019), 1–13. 3

[BKCO16] BELLINI R., KLEIMAN Y., COHEN-OR D.: Time-varying weathering in texture space. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. 2

[BPC16] BONNEEL N., PEYRÉ G., CUTURI M.: Wasserstein barycentric coordinates: histogram regression using optimal transport. *ACM Transactions on Graphics 35*, 4 (2016), 1–10. 3

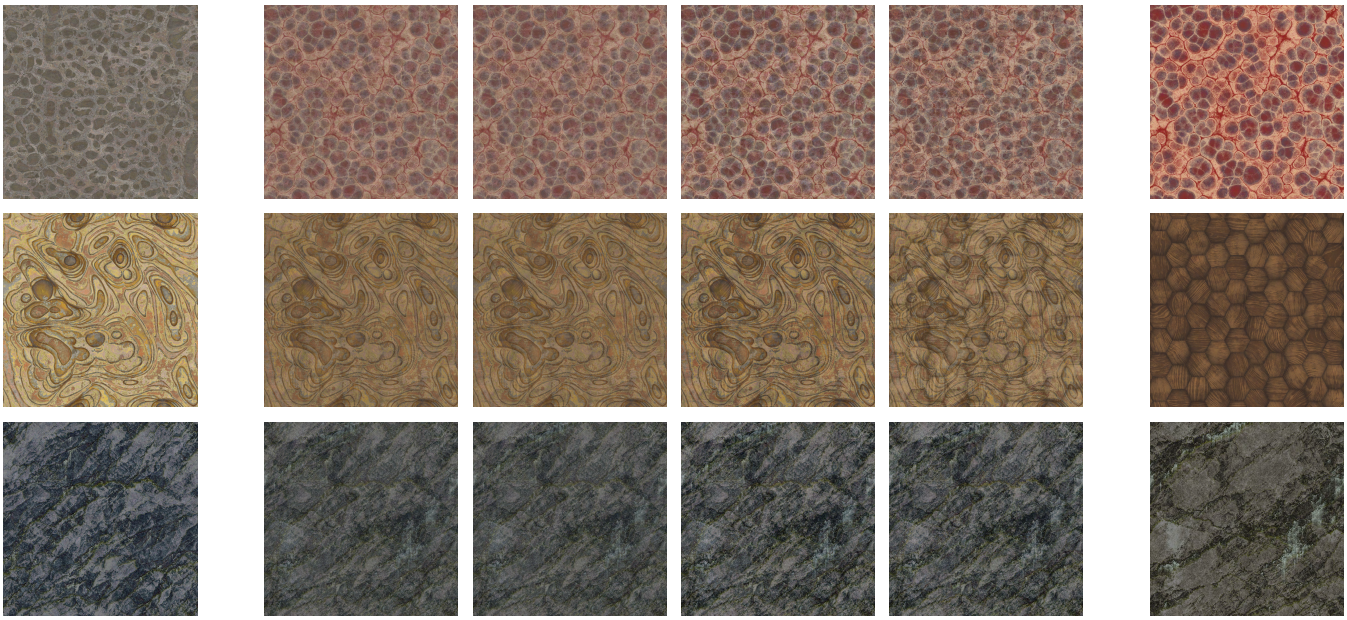

Figure 13: ***Comparison of linearly interpolated RGB texture enhancements.*** *Halfway per-pixel linear interpolations between texture A (leftmost column) and B (rightmost column) are shown. [MZD05] sharpness preservation fails to enhance details on 1K pairs of images and helps to sharpen the result at the pixel scale (require zooming-in). Simpler histogram matching of interpolated textures produces sharp results but has a higher runtime complexity than texture gaussianization, which helps to preserve details after linear blending. Note that no warping is done for this comparison.*

[BRPP15] BONNEEL N., RABIN J., PEYRÉ G., PFISTER H.: Sliced and Radon Wasserstein Barycenters of Measures. *Journal of Mathematical Imaging and Vision 51*, 1 (2015), 22–45. 3

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics 28*, 3 (2009), 1–11. 2

[BTP13] BOUAZIZ S., TAGLIASACCHI A., PAULY M.: Sparse iterative closest point. In *Computer graphics forum* (2013), vol. 32, Wiley Online Library, pp. 113–123. 4

[Bur12] BURLEY B.: Physically-Based Shading at Disney. 26. 1, 8

[Bur19] BURLEY B.: On Histogram-preserving Blending for Randomized Texture Tiling. *Journal of Computer Graphics Techniques (JCGT) 8*, 4 (2019), 8. 2, 6

[BVDPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (2011), pp. 1–12. 3

[BZ17] BARNES C., ZHANG F.-L.: A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media 3*, 1 (2017), 3–20. 2

[CCK*18] CHOI Y., CHOI M., KIM M., HA J.-W., KIM S., CHOO J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8789–8797. 3

[DAD*18] DESCHAINTRE V., AITTALA M., DURAND F., DRETTAKIS G., BOUSSEAU A.: Single-image SVBRDF capture with a rendering-aware deep network. *ACM Transactions on Graphics 37*, 4 (2018), 1–15. 3

[DRCP14] DI RENZO F., CALABRESE C., PELLACINI F.: Appim: Linear spaces for image-based appearance editing. *ACM Trans. Graph. 33*, 6 (Nov. 2014). 2, 7, 8

[DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image melding: combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics 31*, 4 (2012), 1–10. 2, 3

[EKP*20] EFFLAND A., KOBLER E., POCK T., RAJKOVIĆ M., RUMPF M.: Image Morphing in Deep Feature Spaces: Theory and Applications. *Journal of Mathematical Imaging and Vision* (2020). 3

[FH07] FANG H., HART J. C.: Detail preserving shape deformation in image editing. *ACM Transactions on Graphics 26*, 3 (2007), 12. 2

[GAD*20] GUEHL P., ALLÈGRE R., DISCHLER J., BENES B., GALIN E.: Semi-procedural Textures Using Point Process Texture Basis Functions. *Computer Graphics Forum 39*, 4 (2020), 159–171. 2

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680. 3

[GRGH20] GUTIERREZ J., RABIN J., GALERNE B., HURTUT T.: On Demand Solid Texture Synthesis Using Deep 3D Networks. *Computer Graphics Forum 39*, 1 (2020), 511–530. 3

[GSH*20] GUO Y., SMITH C., HAŠAN M., SUNKAVALLI K., ZHAO S.: Materialgan: Reflectance capture using a generative svbrdf model. *ACM Trans. Graph. 39*, 6 (2020), 254:1–254:13. 3, 10

[GvGR16] GROMPONE VON GIOI R., RANDALL G.: Unsupervised Smooth Contour Detection. *Image Processing On Line 5* (2016), 233–267. 3, 4

[HMR20] HENZLER P., MITRA N. J., RITSCHEL T.: Learning a Neural

3D Texture Space From 2D Exemplars. In *CVPR* (2020), pp. 8353–8361. 3

[HN18] HEITZ E., NEYRET F.: High-Performance By-Example Noise using a Histogram-Preserving Blending Operator. *Proc. ACM Comp. Graph. and Int. Tech. 1*, 2 (2018), 1–25. 2, 6

[Kar13] KARIS B.: Real Shading in Unreal Engine 4. 59. 1

[KFG09] KARNI Z., FREEDMAN D., GOTSMAN C.: Energy-Based Image Deformation. *Computer Graphics Forum 28*, 5 (2009), 1257–1268. 2

[KPRN11] KABUL I., PIZER S. M., ROSENMAN J., NIETHAMMER M.: An optimal control approach for texture metamorphosis. *Computer Graphics Forum 30*, 8 (2011), 2341–2353. 2

[Lév15] LÉVY B.: A numerical algorithm for l2 semi-discrete optimal transport in 3d. *ESAIM: Mathematical Modelling and Numerical Analysis 49*, 6 (2015), 1693–1715. 3

[LWX*09] LIU Y., WANG J., XUE S., TONG X., KANG S. B., GUO B.: Texture splicing. *Computer Graphics Forum 28*, 7 (2009), 1907–1915. 2

[MÍ1] MÉRIGOT Q.: A Multiscale Approach to Optimal Transport. *Computer Graphics Forum 30*, 5 (2011), 1583–1592. 3

[Mik19] MIKKELSEN M. S.: Surface Gradient Based Bump Mapping Framework. 24. 7

[MJH*17] MORITZ J., JAMES S., HAINES T. S., RITSCHEL T., WEYRICH T.: Texture Stationarization: Turning Photos into Tileable Textures. *Computer Graphics Forum 36*, 2 (2017), 177–188. 2

[MPTAVG17] MANINIS K.-K., PONT-TUSET J., ARBELÁEZ P., VAN GOOL L.: Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE transactions on pattern analysis and machine intelligence 40*, 4 (2017), 819–833. 3

[MSZM19] MECHREZ R., SHECHTMAN E., ZELNIK-MANOR L.: Saliency driven image manipulation. *Machine Vision and Applications 30*, 2 (2019), 189–202. 2

[MZD05] MATUSIK W., ZWICKER M., DURAND F.: Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 787–794. 2, 3, 5, 6, 8, 10, 11, 12

[NG19] NADER G., GUENNEBAUD G.: Instant transport maps on 2D grids. *ACM Transactions on Graphics 37*, 6 (2019), 1–13. 3

[PP11] PAPARI G., PETKOV N.: Edge and line oriented contour detection: State of the art. *Image and Vision Computing 29*, 2-3 (2011), 79–103. 3

[RGSS10] RUBINSTEIN M., GUTIERREZ D., SORKINE O., SHAMIR A.: A comparative study of image retargeting. In *ACM SIGGRAPH Asia* (2010), p. 1. 2

[RLW*09] RAY N., LEVY B., WANG H., TURK G., VALLET B.: Material Space Texturing. *Computer Graphics Forum* (2009). 2

[RPDB12] RABIN J., PEYRÉ G., DELON J., BERNOT M.: Wasserstein Barycenter and Its Application to Texture Mixing. In *Scale Space and Variational Methods in Computer Vision*, Bruckstein A. M., ter Haar Romeny B. M., Bronstein A. M., Bronstein M. M., (Eds.), vol. 6667. Springer Berlin Heidelberg, 2012, pp. 435–446. Series Title: Lecture Notes in Computer Science. 3

[RSK10] RUITERS R., SCHNABEL R., KLEIN R.: Patch-based texture interpolation. *Computer Graphics Forum 29*, 4 (2010), 1421–1429. 3

[SCO17] SENDIK O., COHEN-OR D.: Deep Correlations for Texture synthesis. *ACM Transactions on Graphics* (2017), 15. 3

[SDGP*15] SOLOMON J., DE GOES F., PEYRÉ G., CUTURI M., BUTSCHER A., NGUYEN A., DU T., GUIBAS L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–11. 3

[SPN*16] SCHMIDT T.-W., PELLACINI F., NOWROUZEZAHRAI D., JAROSZ W., DACHSBACHER C.: State of the art in artistic editing of appearance, lighting and material. *Computer Graphics Forum 35*, 1 (2016), 216–233. 2

[SS09] SHAMIR A., SORKINE O.: Visual media retargeting. In *ACM SIGGRAPH ASIA 2009 Courses on - SIGGRAPH ASIA '09* (2009), ACM Press, pp. 1–13. 2

[STSK20] SCHUSTER K., TRETTNER P., SCHMITZ P., KOBBELT L.: A three-level approach to texture mapping and synthesis on 3d surfaces. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 3*, 1 (2020), 1–19. 2

[UGP*17] UPCHURCH P., GARDNER J., PLEISS G., PLESS R., SNAVELY N., BALA K., WEINBERGER K.: Deep Feature Interpolation for Image Content Changes. In *CVPR)* (2017), IEEE, pp. 6090–6099. 3

[VDKCC20] VACHER J., DAVILA A., KOHN A., COEN-CAGLI R.: Texture interpolation for probing visual perception. In *Advances in Neural Information Processing Systems* (2020), vol. 33, pp. 22146–22157. 3

[WBW20] WEISS S., BAYER F., WESTERMANN R.: Triplanar Displacement Mapping for Terrain Rendering. *Eurographics 2020 - Short Papers* (2020), 4 pages. 7

[WCPW*08] WANG R., CHESLACK-POSTAVA E., WANG R., LUEBKE D., CHEN Q., HUA W., PENG Q., BAO H.: Real-time editing and relighting of homogeneous translucent materials. *The Visual Computer 24*, 7-9 (2008), 565–575. 2

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the Art in Example-based Texture Synthesis. 25. 2

[Wol99] WOLBERG G.: Image morphing: A survey. *The Visual Computer 14* (03 1999). 2

[WTL*06] WANG J., TONG X., LIN S., PAN M., WANG C., BAO H., GUO B., SHUM H.-Y.: Appearance manifolds for modeling time-variant appearance of materials. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 754–761. 2

[WTSL08] WANG Y.-S., TAI C.-L., SORKINE O., LEE T.-Y.: Optimized scale-and-stretch for image resizing. *Proceedings of ACM SIGGRAPH Asia* (2008), 1–8. 2

[XFPA14] XIA G.-S., FERRADANS S., PEYRÉ G., AUJOL J.-F.: Synthesizing and Mixing Stationary Gaussian Texture Models. *SIAM Journal on Imaging Sciences 7*, 1 (2014), 476–508. 3

[YBS*19] YU N., BARNES C., SHECHTMAN E., AMIRGHODSI S., LUKAC M.: Texture mixer: A network for controllable synthesis and interpolation of texture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 12164–12173. 3

[ZKSE16] ZHU J.-Y., KRÄHENBÜHL P., SHECHTMAN E., EFROS A. A.: Generative visual manipulation on the natural image manifold. In *European conference on computer vision* (2016), Springer, pp. 597–613. 3

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *ICCv* (2017), IEEE, pp. 2242–2251. 3

[ZSL*17] ZHOU Y., SHI H., LISCHINSKI D., GONG M., KOPF J., HUANG H.: Analysis and Controlled Synthesis of Inhomogeneous Textures. *Computer Graphics Forum 36*, 2 (2017), 199–212. 2