Eurographics Symposium on Rendering (2024)
E Garces and E. Haines (Guest Editors)

Precomputed Dynamic Appearance Synthesis and Rendering

Yaoyi Bai[1], Miloš Hašan[2], Ling-Qi Yan[1]

[1]University of California, Santa Barbara, USA
[2]Adobe Research, USA

Abstract

Interpolation between objects of varying dimensionality is a common task in computer graphics; however, high-quality dynamic natural interpolation for appearance remains scarce. In this paper, we propose a blending framework for general appearances that can be integrated into renderers without modifying the rendering pipeline. For natural interpolation calculations, we use the mathematical tool optimal transport (OT), known for its promising blending quality. Although recent advancements in OT theory have improved computational performance, integrating runtime OT calculations into the path tracing rendering pipeline compromises algorithm efficiency and increases storage requirements. To address this, we propose a novel solution that precomputes appearances into a proxy distribution and introduces a hierarchical query structure. This enables efficient online point or range data querying, allowing for the generation or retrieval of large data sets as needed. Additionally, the proxy and hierarchical query structure facilitate multi-way barycenter computation. With this efficient query structure and barycentric calculation, we demonstrate several applications of our method, including 2D and 3D interpolation, as well as isotropic BRDF interpolation.

Keywords: appearance synthesis, proxy, query

CCS Concepts

• **Computing methodologies** → **Rendering;**

1. Introduction

Interpolation of varying dimensionality is a common task in computer graphics; valid inputs include 2D and 3D shapes, textures, volumes, BRDFs, color histograms, and many other types of data. However, simple linear blending operations often produce ghosting artifacts, and it is rare to have precise correspondences between features of the objects being interpolated. Even with a high-quality interpolation method, it may still fall short of practical application needs in computer graphics, which demand high performance and additional capabilities, such as the ability to *point-sample* and/or *importance-sample* the interpolated objects without fully constructing them. Point-sampling textures or volumes and importance sampling BRDFs are critical operations for successful integration into real rendering systems.

To avoid ghosting artifacts, we introduce the theory of optimal transport (OT), which offers a promising solution for designing high-quality interpolation methods due to its ability to produce natural interpolated barycenters between distributions [Vil03]. OT has garnered increased attention in the graphics community, due to a sequence of works demonstrating interpolation results scalable to non-trivial instance sizes [BvdPPH11, SdGP*15]. The OT theory defines the Wasserstein distance between two objects, typically formalized as probability distributions or densities. The key step is to optimize a transportation plan to minimize the work of carrying mass from one distribution to another, resulting in natural mass transport. Several techniques exist to accelerate this computation [SdGP*15, BC19, PBC*20], typically by introducing some form of regularization. However, computing barycenters remains computationally expensive, and no existing methods support point queries or spatially varying interpolation weights.

Our ultimate goal is to integrate a general, efficient, and cross-dimensional natural appearance synthesis and rendering solution into traditional renderers. Though OT provides blending results with high fidelity, several additional requirements must be satisfied. Firstly, based on the Monte Carlo *point-sampling* structure of renderers, interpolation or blending operations should work with *discrete point sets*, which also requires *one-to-one* mapping. Moreover, rendering is already a computationally intensive process, so introducing a lightweight blending algorithm is essential. Unfortunately, most of the accelerated OT techniques still cannot meet the performance requirements. Finally, the efficiency and storage constraints limit the possibility of generating a complete set of appearance data during rendering. Therefore, an efficient query structure that can directly return the value of any single point efficiently is essential.

In this paper, we introduce a *proxy distribution* to our pipeline and precompute OT from the appearance distribution to a proxy. The proxy can be any simple distribution, such as uniform or Gaus-

sian. During the precomputation, mappings from the original distribution to the proxy are calculated. At runtime, these mappings are efficiently queried, separating the OT calculation from the rendering process and saving computation time. Additionally, a proxy distribution enables interpolation (barycenter computation) between K distributions. To further accelerate runtime efficiency, a hierarchical query structure is introduced, allowing for the retrieval of each point's value without querying the entire dataset. Finally, our method can be easily integrated into the renderer, enabling swift data queries and barycenter calculations.

Here, we primarily focus on using existing OT solvers to better fit renderers and explore more appearance synthesis applications, rather than introducing novel OT solutions. Considering the structure and Monte Carlo properties of renderers, we prefer discrete OT solvers, with or without regularization, that calculate one-to-one distribution mappings across different dimensions. Based on these criteria, we chose the recent work on Sinkhorn divergences by Feydy et al. [FSV*19] and the fast GPU-based implementation of their approach in the GeomLoss PyTorch library [Geo19]. Sinkhorn divergences are a modification of regularized Wasserstein distances, offering several additional desirable properties that reduce blurring and artifacts compared to other regularization approaches. In summary, our contributions include:

1. a novel, efficient, and general appearance synthesis framework that enhances various rendering applications and can be easily integrated with renderers;
2. a new pipeline that uses optimal transport (OT) to precompute natural mappings between input distributions and a common proxy distribution;
3. a hierarchical point or range query structure for OT-integrated rendering calculations.

2. Related Work

2.1. Appearance Synthesis and Blending

Appearance synthesis has always been an attractive topic in computer graphics. Texture synthesis and bidirectional texture function (BTF) synthesis, along with their applications, have been extensively studied in computer vision and computer graphics [WL00, WL01, TZL*02, RPDB12, LH06, MMS*05]. Recently, neural networks have emerged as another solution for appearance synthesis and interpolation. For example, Sztrajman et al. encoded MERL materials [MPBM03] and interpolated them to generate new materials [SRRW21]. Zsolnai-Feh'er et al. proposed a learning-based system using Gaussian process regression to perform material synthesis, recommending novel materials to artists [ZFWW18]. Fan et al. extended neural blending to layered materials [FWH*22], while Rainer et al. performed nonlinear interpolation on BTF [RJGW19]. Neural networks provide a general and straightforward solution to appearance synthesis; however, these methods often result in blurry outputs due to the high compression rate.

2.2. Optimal Transport in Computer Graphics and Rendering

Optimal transport (OT) has a long history within mathematics, possibly starting with Gaspard Monge in 1781 [Mon81], who formalized the Monge map problem, searching for a mapping between inputs and outputs that minimizes the moving energy. Later, mathematician Leonid Kantorovich defined the modern version of optimal transport as a minimization of a transportation plan from inputs to outputs [Kan42].

The application of Optimal Transport (OT) across various fields in computer graphics and rendering has demonstrated versatility and profound impacts. Bonneel et al. summarized a series of OT applications, such as texture interpolation, sampling, reflectance manipulation, etc. [BD23]; and Peyré et al. surveyed the solvers to compute OT [PC*19]. In texture synthesis, Matusik et al. utilized 1D OT for blending textures [MZD05], while Nadar et al. explored continuous OT maps in 2D [NG18]. Heeger et al. innovatively created textured 3D objects using human texture perception models to avoid distortions typical in texture mapping [HB95]. For BRDF blending, Bonneel et al. [BvdPPH11] applied displacement interpolation, showing impressive interpolation results but still relying on relatively expensive, exact linear programming solutions. Later work by Bonneel et al. [BPC16] introduced Wasserstein barycentric coordinates; this method is more focused on the better fitting of histograms using optimal transport, rather than fast barycenter computation like our work. Cuturi pioneered entropic regularization, presenting a fast, approximate solution through iterative Sinkhorn updates [Cut13]. Solomon et al. extended this methodology to computer graphics, enabling scalable barycenter computation between large discretized distributions [SdGP*15], albeit at the expense of some blurriness in the results. Further, OT has also been applied to BSDF measurement [WKB14]. Apart from texture and BRDF blending, Slice Optimal Transport (SOT) [PBC*20] generates arbitrary dimension of sample distribution to improve the accuracy of Monte Carlo integration, Salaün et al. extended SOT to scalable multi-class sampling [SGSS22]. Bonneel et al. applied Sliced Partial Optimal Transport (SPOT) [BC19] to color transfer and point cloud mapping, and OT also allows blue noise generation [DGBOD12]. Bai et al. precomputed BRDF sample mapping using optimal transport and used lightweight neural networks to compress the mapping for more efficient runtime use [BWZ*22], while Lavenant et al. enabled interpolation over discrete surfaces, showcasing OT's broad applicability [LCCS18].

While our approach shares similarities with linear optimal transport (LOT) [WSB*13], there are distinct differences between them. LOT can easily compute pairwise distance metrics within a large image database, and it allows for using pixel intensities as the particle density. It first calculates an estimated template image and then calculates OT based on a linearized version of the Kantorovich-Wasserstein OT distance. In contrast, our method operates on general appearances across different dimensions rather than being limited to 2D images. Importantly, we leverage OT purely as a tool for providing natural transitions. We do not impose constraints on the choice of OT solver or distance metric, as long as the solver aligns with our renderer integration requirements. Our primary focus remains on utilizing the natural blending capabilities OT offers. Additionally, our proxy distribution is a common and straightforward distribution, such as Gaussian, instead of requiring a newly calculated template.

3. Background

3.1. Appearances and Appearance Synthesis

In physically-based rendering, appearances illustrate the light-scattering properties of materials, commonly represented through textures, volumes, and bidirectional scattering distribution functions (BSDFs).

Consequently, appearance synthesis is often employed in rendering to create additional materials by blending and interpolating existing appearances [DRS10]. To make the synthesized materials as photo-realistic as the original ones, simple linear blending or interpolation methods are tricky since they result in ghosting artifacts, including most of the neural solutions. Therefore, a strategy that ensures natural interpolation and preserves the intricate details of the original appearances is essential. To address this, we employ optimal transport (OT), renowned for facilitating natural interpolations, to enhance our approach to appearance synthesis.

3.2. Optimal Transport and Barycenters

In this subsection, we briefly review the concepts and properties of optimal transport, particularly its natural blending property. Many treatments of the topic in the applied mathematics literature model the transported quantities as probability distributions or measures on abstract domains, which could be continuous or discrete; this approach provides maximal generality. However, to better fit the Monte Carlo framework of the path tracing rendering pipeline, we focus on discretized point distributions at any dimension and one-to-one mappings.

To be more specific, we focus on a Lagrangian approach to approximate optimal transport, treating the points as particles in a d-dimensional Euclidean space whose positions can move, as opposed to the Eulerian approach of fixing a discretization of a domain and solving for the values of the distribution at the predetermined grid points, like in Solomon et al. [SdGP*15]. While neither of these approaches is universally superior, we choose the Lagrangian approach to define a fast point query for better renderer integration.

Point distributions. We define a point distribution $\boldsymbol{\alpha}$ with n points as

$$\boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \delta_{x_i}, \tag{1}$$

where the $x_i \in \mathbb{R}^d$ are the sample positions, α_i are the probabilities such that $\sum_i \alpha_i = 1$, and the δ_{x_i} are Dirac delta impulses at positions x_i, intuitively representing units of mass that are infinitely concentrated at locations x_i.

Similarly, we define a second point distribution $\boldsymbol{\beta}$, with n samples at positions $y_i \in \mathbb{R}^d$ and probabilities β_i summing to one. In our specific situation, we constrain the transported point distributions to have the same number of points n to fit the Monte Carlo path tracing framework.

The particles can carry other quantities (colors, weights, reconstruction kernel widths, etc.), which do not participate in the optimal transport computation, but affect how the final result is used in a graphics application. A visualization of discrete point mapping and movement is shown in Figure 1.

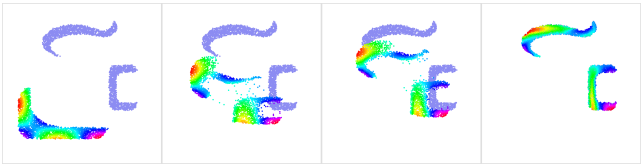


Figure 1: Visualizations of point distribution $\boldsymbol{\alpha}^m$ (color-coded to show the movement of particles) and $\boldsymbol{\beta}$ (purple) after $m = 0$, 1, 2 and 10 iterations from left to right using the GeomLoss PyTorch library [Geo19].

Wasserstein distances and barycenters. Define the Wasserstein distance $\mathcal{W}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ between two such point distributions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as the cost $\sum_{ij} \pi_{ij}$ of the optimal transportation plan π. For many practical applications in computer graphics, the pairwise squared distance has nice properties and physical interpretation [BD23].

Now we can define the k-way Wasserstein barycenter $\mathcal{B}$ between point distributions $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_k$ as the point distribution $\boldsymbol{\beta}$ that minimizes a convex combination of Wasserstein distances from β to the input distributions, and $w_1, \cdots, w_k$ are weights that sum to one:

$$\mathcal{B}(\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_k) = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{k} w_i \mathcal{W}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}). \tag{2}$$

Wasserstein barycenters often provide plausible interpolations of the input point clouds. For example, given two input Gaussian distributions with different means and standard deviations, Wasserstein barycenter interpolation will naturally interpolate their means and shapes. In contrast, linear blending would result in a simplistic solution, with both distributions merely changing in weight without shifting positions.

4. Our Method

With the high-quality blending results using squared distance cost and the notable capability of the optimal transport framework for multi-way interpolation [BD23], we introduce our method of precomputed appearance synthesis with the help of OT in this section.

4.1. Our Pipeline

Precomputation. Before the rendering process, we calculate the optimal transport from a proxy distribution $\boldsymbol{\beta}$ to a target input distribution $\boldsymbol{\alpha}_i$, as shown in Figure 2. Note that the figure illustrates 2D textures, but there is no dimensional constraint on the distributions. The proxy now serves as the common distribution, containing mapping connections to all the other input distributions. Since OT calculation is time-consuming, performing it during the precomputation stage alleviates the extra performance burden on the renderer.

Runtime. During the rendering process, the renderer uses the precomputed proxy-to-input OT mappings for appearance synthesis. However, to accurately illustrate the appearances, an enormous

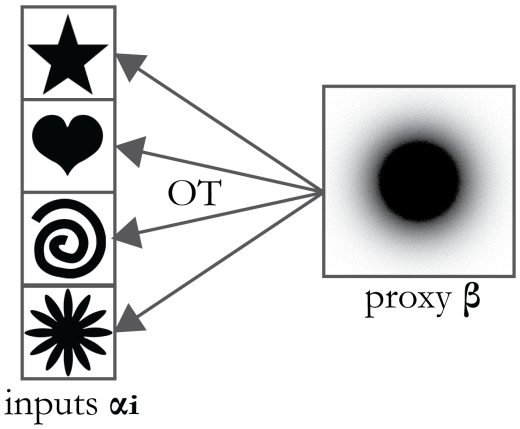


Figure 2: During the precomputation, optimal transport is calculated from a common proxy distribution $\boldsymbol{\beta}$ to all the input target distributions $\boldsymbol{\alpha}i$, where i stands for the i-th input. Note that this figure only shows a 2D texture as an example, and we do not constrain the appearance dimensions.

number of samples are required, and sometimes millions of samples need to be stored and looked up. Therefore, before the per-pixel rendering process, we build a hierarchical structure for faster data queries, which we will explain in Section 4.2.

Proxy selection. For the generalization of our method, we do not constrain any distribution as the "optimal" one. Figure 3 shows the 2D texture synthesis results using different distributions as proxies. The results show that using different proxies causes subtle visual differences, which are acceptable for rendering purposes since natural transitions and blending are already guaranteed by optimal transport.

4.2. 2-way and K-Way Interpolation with Query

Fast 2-Way Interpolation Without Proxy Consider the problem of computing the barycenter at an interpolation point u between distributions $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$. Having established the approximate two distributions $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$, we can compute the barycenter $\boldsymbol{\beta}$ at an interpolation point u by simply linearly interpolating the positions for **all** corresponding pairs of points:

$$y_i = (1-u)x_i^1 + ux_i^2. \tag{3}$$

However, note that this is only correct barycenter if the distance cost function is Euclidean; otherwise, this approach will give plausible blends that are not equivalent to barycenters. This fast 2-way interpolation method contributes to the last row in Figure. 3.

The query structure. Now we introduce a variation of the above 2-way interpolation algorithm that supports *point query* or *range query*. Instead of having to fully construct the point distribution at an interpolation point u, we can query the value of the final continuous reconstructed distribution at interpolation point u and a spatial point $p \in \mathbb{R}^d$ in logarithmic time relative to the number of samples n. This is useful for many graphics algorithms that need to query the result point-wise or range-wise, such as implicit surface or volume ray tracing algorithms applied to the resulting distribution. To

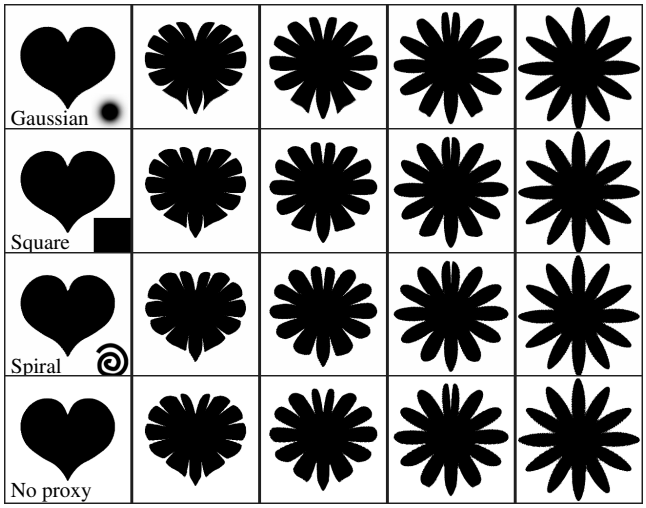


Figure 3: Comparisons on different choices of proxies. The first three rows show our approximate optimal transport results using a Gaussian proxy, a square proxy, and a spiral line-shaped proxy, respectively. The last row is computed by linearly interpolating the Monge map between two inputs directly. As seen, the differences in using different proxies are subtle. Note again that our method works on discrete point clouds and we reconstructed the results shown in this figure.

achieve such a sub-linear point query capability, we need an acceleration structure that guides the search for the nearest particles for any u and any p.

We achieve this by building a hierarchical structure on top of the particle positions $\mathbf{x}$ of $\boldsymbol{\alpha}i$ (where i stands for the i-th input) inspired by bounding volume hierarchy (BVH). Our BVH is similar to a traditional BVH tree. Each node contains two axis-aligned AABB bounding boxes, Ai_j and B_j (where j stands for hierarchical level), for distribution $\boldsymbol{\alpha}i$ and proxy $\boldsymbol{\beta}$. Additionally, each node includes all samples in $\boldsymbol{\alpha}i$ and $\boldsymbol{\beta}$, and two child pointers, as shown in Figure 4. When constructing the tree, we always calculate the splitting dimension based on the longest Euclidean distance of samples in one of the dimensions in $\boldsymbol{\beta}$ and divide the proxy bounding box B_j in half along this dimension until only one sample remains. The OT-mapped samples in $\boldsymbol{\alpha}i$ are assigned to one of the child nodes based on their corresponding proxy sample division, allowing us to calculate the input bounding box Ai_j. This design enables us to quickly map a query location by looking it up in the divided proxy and facilitates easier calculations for range queries.

At runtime, our query function takes interpolation point u and a spatial point $p \in \mathbb{R}^d$. To use the BVH, our point query compares the query location with the proxy bounding box B_j until it traverses to the leaf node. Note that the node already contains a matching sample location from $\boldsymbol{\alpha}i$. Similarly, we perform the same traversal on the same query location in different input trees to find the corresponding input samples, keeping the proxy consistent as shown in Figure 4. Finally, blending can be performed on these input samples.

Point query and range query. Apart from the leaf nodes, each

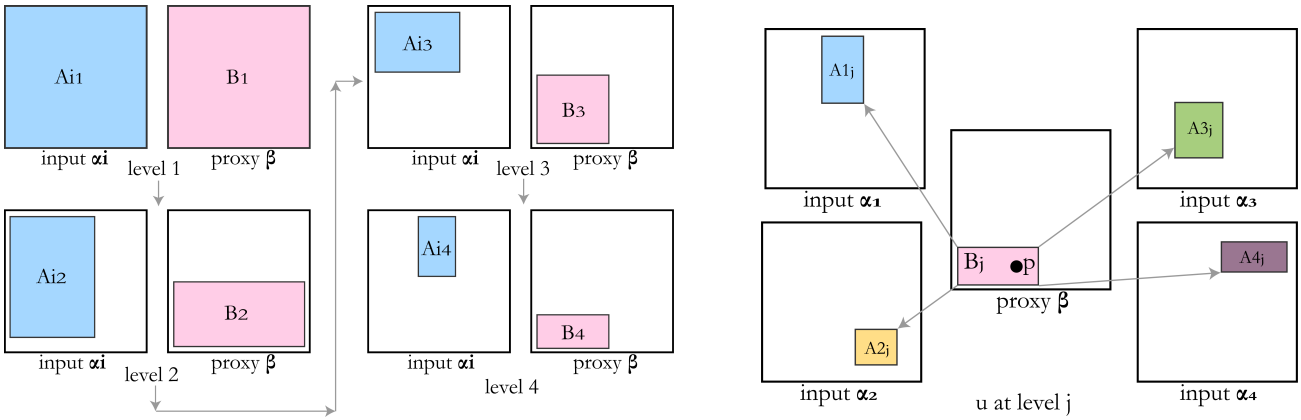


Figure 4: Visualization of our hierarchical structure. (Left) The bounding box subdivision at each level. OT ensures that each sample in the blue bounding box Ai_j will be mapped to a corresponding sample in the pink bounding box B_j. (Right) For a four-way blending, we start by querying the spatial point p in B_j from $\boldsymbol{\beta}$ at level j and find the corresponding mapped location from the four inputs in Ai_j from $\boldsymbol{\alpha i}$. The result will be obtained by blending the bounding boxes of the four inputs.

bounding box of the hierarchical structure in Figure 4 naturally holds multiple point samples. In this case, our hierarchical structure is performing range query, while point query is a special case in which only leaf bounding boxes are looked up.

Finally, note that we can easily compute the bounding boxes for the distribution at any interpolation point u, by simply linearly blending the vertices of corresponding boxes Ai_j and B_j. Indeed, as the particles travel along straight lines, the blended bounding boxes are guaranteed to cover the interpolated point distributions. Therefore, given this hierarchical structure can quickly produce bounding boxes for any interpolated distribution, we can execute any kind of nearest neighbor search for the interpolated point cloud at point u in Figure 4, without constructing any data structures specific to u.

5. Implementation Details

5.1. Data Preparation

Sampling input distributions. Since our approach works on point distributions, it is important that the sample points accurately capture the original continuous input distributions. We convert the inputs into point clouds by importance sampling. In many cases, the input distributions are defined on a regular grid (for example, 2D images on pixels and 3D volumes/BRDFs discretized on 3D grids). Assuming these are distributions, then the grids record the probability density functions (pdf-s).

Importance sampling of such d-dimensional discretized pdfs on regular grids is a well-studied problem, common in applications such as sampling a 2D environment map. Let us assume a 2D grid. First, we calculate a cumulative distribution function (cdf) for each row, then we marginalize the 2D data along each row to get a 1D column vector and calculate the cdf for this column. For importance sampling, we first importance sample a row from the marginalized column, then importance sample the selected row using its cdf to find a sample column. This approach easily extends to higher dimensions. To guarantee our samples are evenly distributed, we draw d-dimensional Sobol pseudo-random numbers in the $[0,1]^d$ space.

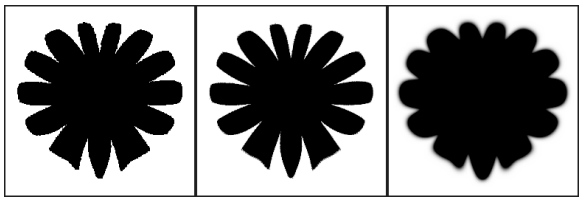


Figure 5: Comparison of different query sizes. (Left) A box with the same size as a pixel. (Middle) A Gaussian with a standard deviation of $1/3$ of a pixel's side length. (Right) A Gaussian with a standard deviation of 4 times a pixel's side length. As can be seen, our choice of Gaussian (middle) creates a similar but more anti-aliased result as compared to be box (left) and is not overblurred as using a larger Gaussian (right).

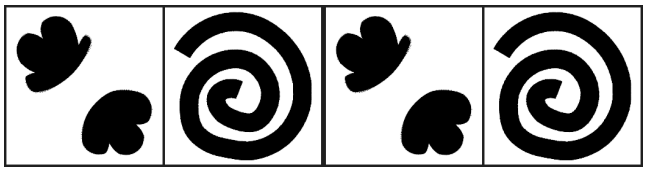


Figure 6: Comparison of the performance between point query (left two images, ~ 0.032s each) and kernel splatting (right two images, ~ 0.004s each). The trade-off between flexibility for rendering and speed for image/model generation has widened the practical use of our method.

One special case is to sample uniformly inside a solid object (water-tight mesh). In this case, we simply draw random 3D points within the object's bounding box and then determine whether each point is inside the object. To do this, we shoot a ray from the sample point towards a random direction and count the number of intersec-

tions found along this ray. An odd number of intersections indicates that the random sample is inside the object, and vice versa.

As a reference, the storage cost of a precomputed 1 million sample 2D texture with all colors, such as the inputs in Figure 9, Figure 12, and Figure 10, is 19 MB. For precomputed 1 million 3D volumes without color, as shown in Figure 13 and Figure 11, as well as the 4D BRDF data (explained further in Section 6), the storage is 11 MB.

5.2. Results Generation and Renderer Integration

Reconstruction. For results outside renderers, reconstruction is required since discrete samples are technically delta functions. To turn the resulting barycenter point cloud into final rendering results, we convert each delta function into a small normalized Gaussian kernel, truncating the Gaussians at (typically) 3 standard deviations. Larger and smaller queries simply trade-off between blurring and noise. For 2D images, we make the size of the kernel query comparable to the pixel size. Figure 5 provides a comparison using different sizes of point queries.

Query and splatting. For applications that do not require point or range queries, the barycenters are explicitly generated, as demonstrated in Figure 6. As expected, the kernel splatting method is much faster for explicitly generating entire barycenters compared to exhaustively performing point queries for every pixel of the result. On the other hand, the point query allows for greater flexibility, larger/smoother queries, and may potentially be more efficient than splatting in higher dimensions, as the number of neighbors increases with dimensionality.

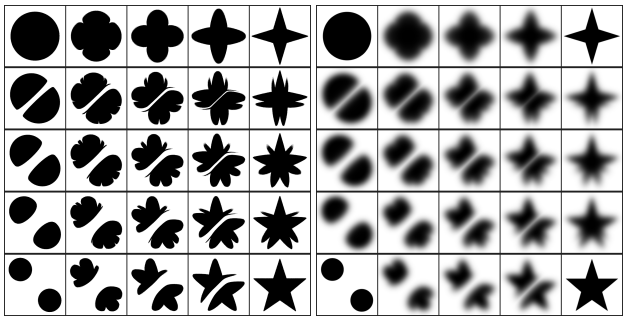
Renderer integration. For all other applications inside the renderer, we integrate our framework into the Mitsuba renderer [Jak10] with minimal revisions, specifically to the evaluation functions within the corresponding appearance classes such as texture, volume, and BRDF. These evaluation functions provide pairs of query locations in the form of texture UV coordinates, locations inside the volume, and pairs of incident and outgoing ray directions. We convert these query locations to a range between 0 and 1 for each dimension, and then look up the precomputed hierarchical structure to obtain the mapping location and color value. Since the query location can only be implicitly provided by these evaluation functions, splatting does **NOT** work inside the renderer.

6. Applications, Results, and Comparisons

In this section, we present practical applications of appearance synthesis and rendering using our framework. We have implemented these applications inside the Mitsuba renderer [Jak10]. All timings in this section are measured on a 3.60GHz Intel i9 CPU (8 cores, hyperthreaded to 16 threads) with 32 GB of main memory.We also used an NVIDIA GTX 1080 GPU, but only for OT precomputation.

6.1. Interpolating 2D Shapes

4-way barycenters in 2D. In Figure 7, we show a 4-way blending of different shapes using our approximate optical transport. In this example, we choose the proxy as a 2D Gaussian with a standard deviation of $1/6$ the side length of the input images. The resolution of the input images is 256×256, so are the generated barycenters. During the precomputation stage, we treat the input images as 2D probability distributions and draw $250K$ points from each input by importance sampling. We present a side-by-side comparison with the convolutional Wasserstein method [SdGP*15], using the implementation in the Python Optimal Transport (POT) library [POT19]. The convolutional Wasserstein method produces blurrier results compared to ours.



(a) Ours: proxy approx. **Pre.: 834.7s** Point.: 7.3s / Splat.: 0.4s (b) Convolutional Wasserstein $\epsilon = 0.005$, 21.2s

Figure 7: 4-way barycenters of different shapes (at the corners) using (a) our approximate optimal transport with a Gaussian proxy, and (b) convolutional Wasserstein. As we can see, the convolutional Wasserstein method produces blurred results and is much slower than our runtime method. Moreover, the precomputation of our method is performed on GPU (marked in bold).

Additionally, the decoupling of the precomputation and runtime computation in our method significantly improves performance. As introduced in Section 4, our method can compute barycenters either fully using kernel splatting or through a point query. The full computation approach yields a $53\times$ performance improvement compared to the convolutional Wasserstein method. If the barycenter is point queried for every pixel, our method is still $2.9\times$ faster than the convolutional Wasserstein method, which computes the entire barycenters at once.

6.2. Interpolating 2D Textures

Apart from interpolating 2D shapes, we would like to explore more 2D texture rendering applications.

Tileable textures interpolation. Tileable textures are another ubiquitous appearance used in rendering and by artists. Since tileable properties allow infinite extension, a natural interpolation strategy will greatly benefit rendering in terms of storage and efficiency optimization. Figure 8 shows a bathroom with a blended marble back wall. The back wall is a 4-way interpolation using the input tileable images shown in Figure 9. Four-way texture interpolation can be easily integrated into the renderer with any input images and our appearance synthesis framework. To further demonstrate the tileable properties of our input images, we created a zigzag blending similar to [MZD05].

Interpolating 2D tileable textures with Perlin noise. Blending

Figure 8: The rendering result of a bathroom with four marble tileable images as the textures at the back. The input images can be found in Figure 9.

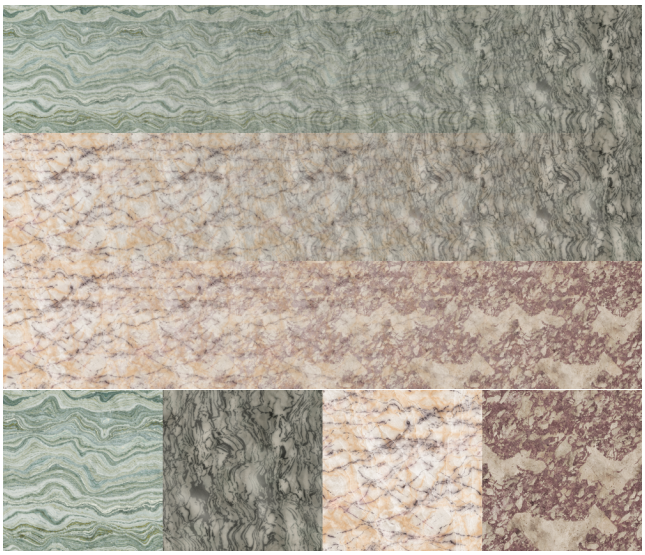


Figure 9: (Top) A zig-zag blending of 4 tileable images. (Bottom) The input tileable 2D textures, generated from Adobe Substance 3D Asset.

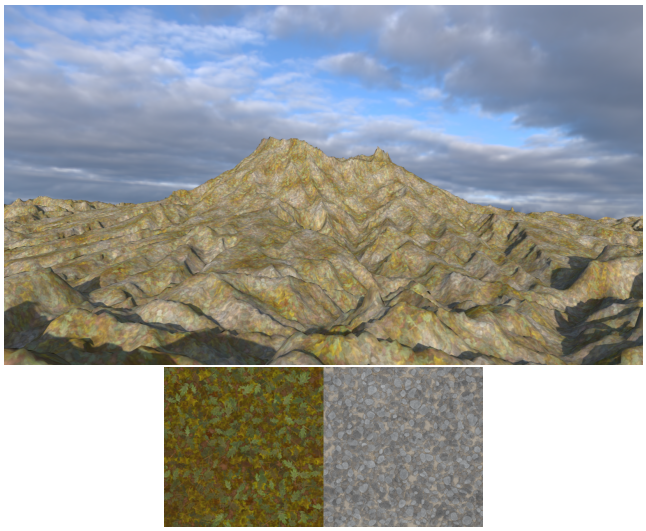


Figure 10: (Top) The rendering result of a mountain using two tileable images as textures and Perlin noise as the blending weight. A video sequence included in the supplementary material shows a flyover of the blended mountain. (Bottom) The input tileable textures are generated from Adobe Substance 3D Asset.

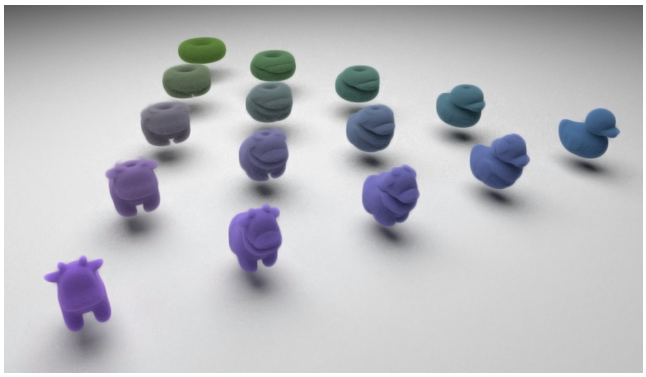


Figure 11: A triangle of interpolated 3D volumes using a 3D Gaussian proxy. All the barycenters, including the three inputs at the three vertices of the triangle, are point queried on the fly without being explicitly generated.

two textures using Perlin noise [Per85] as the interpolation weight can create smooth and infinitely extendable results. Inspired by these properties of Perlin noise, we extend the previous tileable texture blending results that merely used linear interpolation to adopt Perlin noise as the blending weight. Figure 10 shows the rendering result of a mountain with leaves and pebbles tileable textures blended. With the tileable textures, Perlin noise as the blending weight, and our appearance synthesis framework, an infinitely extensive terrain with two textures can be easily generated, with the storage cost of only two precomputed OT hierarchical structures.

Range Query and Level of Detail. In computer graphics, level of detail (LOD) refers to the technique where the complexity of a model, texture, or other metrics such as object importance and viewpoint-relative speed or position decreases as the model moves away from the viewer. This is because rendering all the details becomes less critical at greater distances. LOD is widely used in the game and animation industry. Introducing the idea of LOD into our framework allows for a significant boost in appearance rendering efficiency and is extensively applied in real-time rendering. To implement LOD, we include a range query in our rendering pipeline that calculates a summed or averaged value over a specified range. Figure 12 shows rendering comparisons between three sets of 2D texture interpolation. While we demonstrate LOD results for 2D

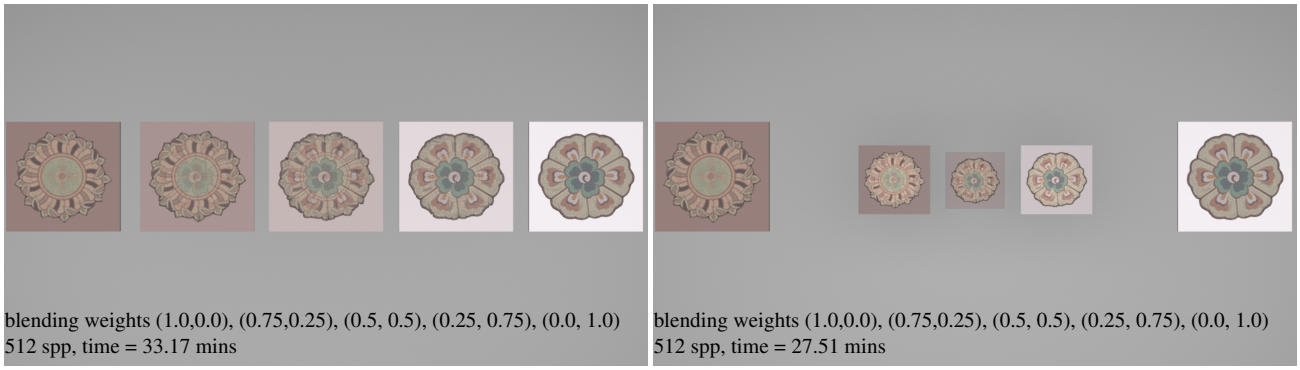


Figure 12: When a model or texture is far from the camera, LoD helps reduce its complexity to save on rendering calculations. We present a comparison of five texture blending results. (Left) All five textures are positioned at an equal distance from the camera. (Right) The middle three textures are moved away from the camera along the view direction. The performance comparisons show a significant decrease in rendering time when the textures are moved further away.

textures, it is important to note that the LOD application of our method does **NOT** have a dimension constraint.

6.3. Interpolating 3D Volume

We now apply our method to blend 3D shapes, which we then render as densities of participating media. We treat each input 3D mesh as a 3D distribution, and importance sample the volume it bounds it with 1 million samples. To compute the k-way correspondences, we choose a 3D Gaussian as the proxy, which is easily importance-sampled due to its separability.

previous optimal transport solutions would need to fully realize each 3D barycenter using optimal transport on the fly and then use it for rendering. However, with the point query option enabled by our method, we no longer need to explicitly generate the 3D barycenters. Instead, we query the density at any point in any barycenter whenever needed by the underlying renderer. In Figure 11, we show an example of 3-way blending among three uniformly sampled models: cow, duck, and torus. Using our point query method, no intermediate 3D volumes need to be generated, thus introducing no additional memory consumption at runtime.

Figure 13 shows another example of a 2-way heterogeneous approximate optimal transport between 3D shapes. In this example, the monkey model is uniformly sampled and the smoke model is importance sampled. We present the blended results at different time steps.

6.4. Interpolating BRDF

Another application of our method is measured BRDF blending. Given k different BRDFs, we aim to find a smooth transition between them. Specifically for rendering, we benefit from a point query for BRDF evaluation for arbitrary pairs of incident and outgoing directions. We use the MERL BRDF database [MPBM03], where all data is purely measured without fitting any analytical representation, and spans a wide range of material types.

The BRDFs in the MERL database are isotropic, and we follow the original 3D $(\theta_i, \theta_o, \phi_d)$ setting from the MERL dataset for more efficient calculation. Previous optimal transport interpolations of BRDFs [SdGP*15] would interpolate 2D slices for a fixed incoming direction, which we could also do; however, our method scales to interpolating BRDFs as entire 3D objects. We first prepare each BRDF as a 3D grid with resolution 180^3 in the $(\theta_i, \theta_o, \phi_d = \phi_o - \phi_i)$ space, reshaped from the original data. Each grid cell stores an RGB component and a corresponding luminance. We also record an average luminance for each BRDF.

With the prepared data, we now use the optimal transport to interpolate the BRDFs. Similar to previous applications, we

1. importance-sample each 3D BRDF with 1 million samples,
2. precompute the optimal transport between the samples from each BRDF and a sampled 3D Gaussian proxy, and
3. during runtime, use point query to find the value of any barycenter at any incoming and outgoing direction.

However, recall that optimal transport traditionally assumes the inputs to be probability distributions and produces barycenters as probability distributions as well. Therefore, the point query is essentially giving us the interpolated BRDF's normalized luminance. We linearly interpolate the per-BRDF average luminance to get the overall luminance normalization constant of the barycenter. After that, we re-introduce color to the BRDF, by tracking the color-to-luminance ratio (an RGB triple) per point. When any point on the implicit barycenter is found, we compute the interpolated color-to-luminance ratio from all the inputs. This ratio is then used to reconstruct the true BRDF value at a query point.

In Figure 14, we show the 4-way blending results of four different BRDFs selected from the MERL database. We also provide a visualization of 2D BRDF slices at $\theta_i = 45^\circ$. As we can see, not only the shapes of different BRDFs are plausibly interpolated, providing a natural glossy-to-diffuse transition, but also the overall luminance and spatially varying colors. Note that the BRDF slices in Figure 14 are meant to be mostly dark, and we have adjusted the curves to make them more visible; thus, the noise level also perceptually increases.

In Figure 15, we show a series of 2-way blending results between two input BRDFs without a proxy. The interpolated BRDFs in be-

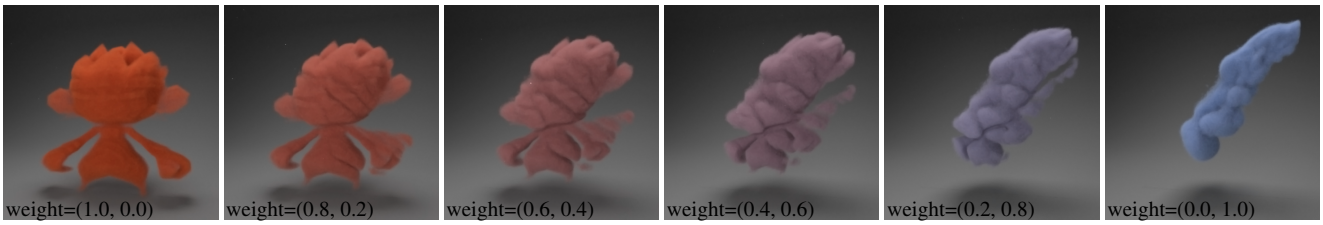


Figure 13: 3D heterogeneous volume interpolation from a monkey model to a smoke model. The images are rendered using a Gaussian proxy, and 1 million point samples. We also include a video in the supplementary material.

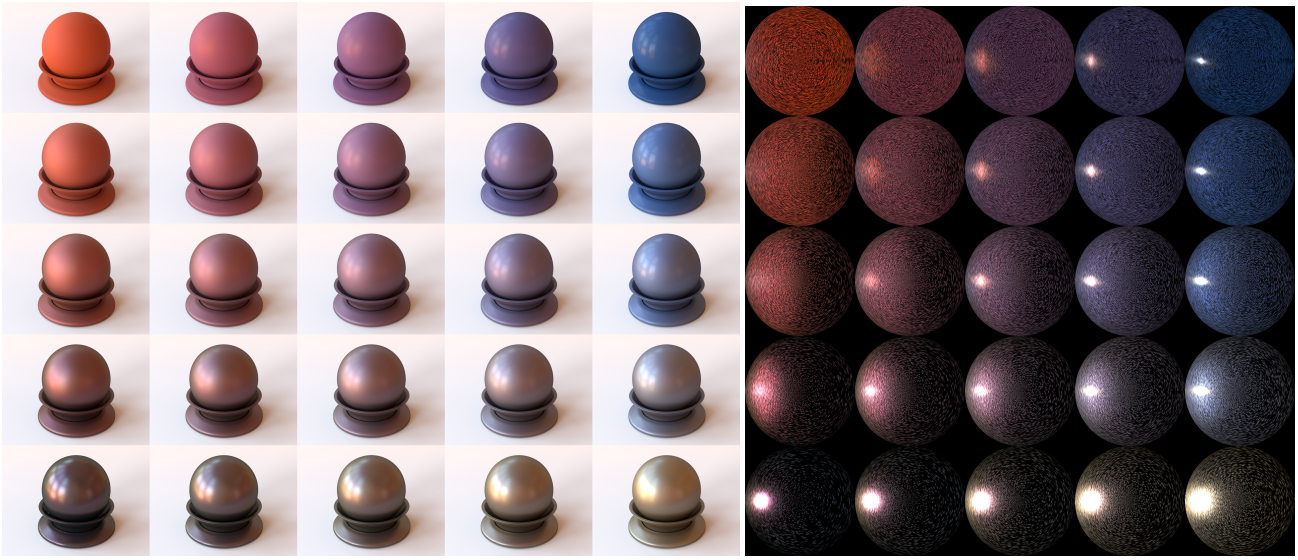


Figure 14: (Left) A 5×5 array of renderings of 4-way blended BRDFs on the *matpreview* scene. The 4 input BRDFs are at the corners and are chosen from the MERL database [MPBM03] as representatives (diffuse, plastic, iridescence, and metallic). For each corner, the blending weight starts at 1.0 for each BRDF and decreases by 0.2 with each step. Note that the diffuse component, the highlight, and the colors are all plausibly blended. (Right) A visualization of corresponding 2D BRDF slices on unit disks with a fixed $\theta_i = 45^\circ$ coming from the right. We also include a video about our BRDF blending transition in the supplementary material.

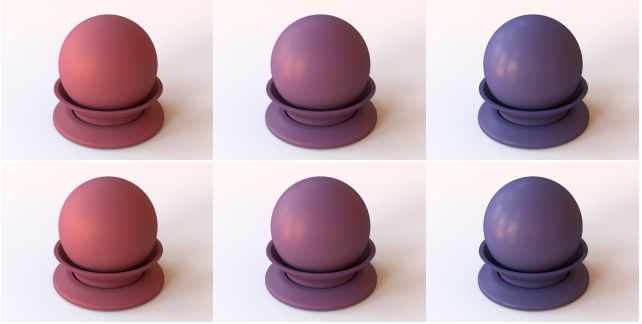


Figure 15: 2-way barycenters of the same two inputs from the first row of Figure 14 (excluding the two inputs themselves), using the proxy (top row) and without proxy (bottom row). Similar to the 2D case, the differences in using these two methods are subtle.

tween are explicitly generated without point queries. The two input BRDFs are the same as the ones in the top-left and top-right corners in Figure 14. Compared to the first row in Figure 14, we find that there's no visible difference, indicating that our proxy method also works well in this task.

For anisotropic BRDFs, we could extend our method to importance-sample the full 4D input BRDF data–the sampling and BVH structure for point query extends to 4D naturally, and the Lagrangian method for optimal transport makes no assumptions on the dimension of the sample points. The limiting factor is the lack of measured data in the MERL database. Again, we could also interpolate per 2D BRDF slice rather than directly in the 4D space.

7. Conclusion and Future Work

In this paper, we leverage recent improvements in the field of optimal transport to enable applications in appearance synthesis. Our techniques convert input distributions into point clouds, scaling easily to one million points. We precompute maps between a proxy distribution and each input distribution, allowing us to compute barycenters almost immediately after an initial precomputation stage. Furthermore, we introduce an efficient point or range query operation, enabling us to query the resulting interpolated objects at points of our choosing. We demonstrate several applications

of this point query operation to 2D and 3D shape interpolation, as well as fully measured BRDF interpolation based on MERL data.

There are challenges not yet solved by our approach. We currently lack a theoretical analysis that could explain the surprisingly good performance of the proxy approach for k-way barycenters. Additionally, we do not have a theoretical understanding of which proxy works best. These are key areas we aim to explore and extend in future work.

While optimal transport usually produces visually plausible results, this is not guaranteed from a human perspective; even "ground truth" barycenters can occasionally contain objectionable artifacts, such as the splitting of features that, from a human perspective, should remain intact. Exploring human-oriented metrics in optimal transport would also be an interesting area of study.

References

[BC19] BONNEEL N., COEURJOLLY D.: Spot: sliced partial optimal transport. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–13. 1, 2

[BD23] BONNEEL N., DIGNE J.: A survey of optimal transport for computer graphics and computer vision. In *Computer Graphics Forum* (2023), vol. 42, Wiley Online Library, pp. 439–460. 2, 3

[BPC16] BONNEEL N., PEYRÉ G., CUTURI M.: Wasserstein barycentric coordinates: Histogram regression using optimal transport. *ACM Trans. Graph. 35*, 4 (July 2016). URL: https://doi.org/10.1145/2897824.2925918, doi:10.1145/2897824.2925918. 2

[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* (New York, NY, USA, 2011), SA '11, Association for Computing Machinery. URL: https://doi.org/10.1145/2024156.2024192, doi:10.1145/2024156.2024192. 1, 2

[BWZ*22] BAI Y., WU S., ZENG Z., WANG B., YAN L.-Q.: Bsdf importance baking: A lightweight neural solution to importance sampling general parametric bsdfs. *arXiv preprint arXiv:2210.13681* (2022). 2

[Cut13] CUTURI M.: Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26*, Burges C. J. C., Bottou L., Welling M., Ghahramani Z., Weinberger K. Q., (Eds.). Curran Associates, Inc., 2013, pp. 2292–2300. 2

[DGBOD12] DE GOES F., BREEDEN K., OSTROMOUKHOV V., DESBRUN M.: Blue noise through optimal transport. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 1–11. 2

[DRS10] DORSEY J., RUSHMEIER H., SILLION F.: *Digital modeling of material appearance*. Elsevier, 2010. 3

[FSV*19] FEYDY J., SÉJOURNÉ T., VIALARD F.-X., AMARI S.-I., TROUVÉ A., PEYRÉ G.: Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), PMLR, pp. 2681–2690. 2

[FWH*22] FAN J., WANG B., HAŠAN M., YANG J., YAN L.-Q.: Neural layered brdfs. In *Proceedings of SIGGRAPH 2022* (2022). 2

[Geo19] Geomloss, 2019. https://www.kernel-operations.io/geomloss/. 2, 3

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 229–238. 2

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 6

[Kan42] KANTOROVICH L.: On the transfer of masses (russian). vol. 37 of *Doklady Akademii Nauk*, pp. 227–229. 2

[LCCS18] LAVENANT H., CLAICI S., CHIEN E., SOLOMON J.: Dynamical optimal transport on discrete surfaces. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–16. 2

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 541–548. 2

[MMS*05] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis, and rendering of bidirectional texture functions. In *Computer Graphics Forum* (2005), vol. 24, Wiley Online Library, pp. 83–109. 2

[Mon81] MONGE G.: Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.* (1781), 666–704. 2

[MPBM03] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: A data-driven reflectance model. *ACM Transactions on Graphics 22*, 3 (July 2003), 759–769. 2, 8, 9

[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, 6

[NG18] NADER G., GUENNEBAUD G.: Instant transport maps on 2d grids. *ACM Transactions on Graphics 37*, 6 (2018), 13. 2

[PBC*20] PAULIN L., BONNEEL N., COEURJOLLY D., IEHL J.-C., WEBANCK A., DESBRUN M., OSTROMOUKHOV V.: Sliced optimal transport sampling. *ACM Trans. Graph. 39*, 4 (2020), 99. 1, 2

[PC*19] PEYRÉ G., CUTURI M., ET AL.: Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning 11*, 5-6 (2019), 355–607. 2

[Per85] PERLIN K.: An image synthesizer. *ACM Siggraph Computer Graphics 19*, 3 (1985), 287–296. 7

[POT19] Python optimal transport, 2019. https://pot.readthedocs.io/en/stable/. 6

[RJGW19] RAINER G., JAKOB W., GHOSH A., WEYRICH T.: Neural btf compression and interpolation. *Computer Graphics Forum (Proceedings of Eurographics) 38*, 2 (Mar. 2019). 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: Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29–June 2, 2011, Revised Selected Papers 3* (2012), Springer, pp. 435–446. 2

[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 Trans. Graph. 34*, 4 (July 2015). 1, 2, 3, 6, 8

[SGSS22] SALAÜN C., GEORGIEV I., SEIDEL H.-P., SINGH G.: Scalable multi-class sampling via filtered sliced optimal transport. *arXiv preprint arXiv:2211.04314* (2022). 2

[SRRW21] SZTRAJMAN A., RAINER G., RITSCHEL T., WEYRICH T.: Neural brdf representation and importance sampling. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 332–346. 2

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (ToG) 21*, 3 (2002), 665–672. 2

[Vil03] VILLANI C.: *Optimal Transport*. Springer-Verlag Berlin Heidelberg, 2003. 1

[WKB14] WARD G., KURT M., BONNEEL N.: Reducing anisotropic bsdf measurement to common practice. In *Material Appearance Modeling* (2014), pp. 5–8. 2

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 479–488. 2

[WL01] WEI L.-Y., LEVOY M.: Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 355–360. 2

[WSB*13] WANG W., SLEPČEV D., BASU S., OZOLEK J., ROHDE G.: A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision 101* (06 2013), 254–269. doi:10.1007/s11263-012-0566-z. 2

[ZFWW18] ZSOLNAI-FEHÉR K., WONKA P., WIMMER M.: Gaussian material synthesis. *arXiv preprint arXiv:1804.08369* (2018). 2