Heat Kernel Textures the Geodesic Gaussians That Do Not Splat

Simone Foti*, Caner Korkmaz*, Stefanos Zafeiriou, and Tolga Birdal

Imperial College London

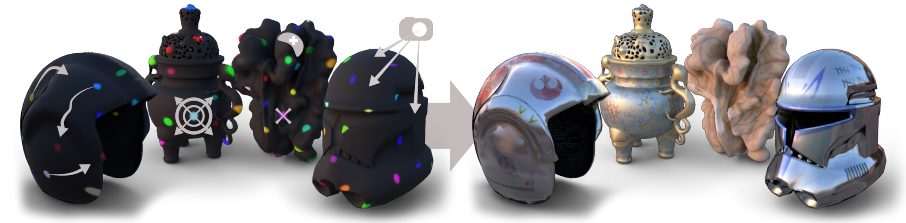


Fig. 1: HKTex: an intrinsic, UV-free texture representation using heat kernels. *Left:* A Riemannian framework constrains kernel movement to the surface, followed by manifold-aware controllers that manage kernel shape and density. Evaluation occurs within a differentiable ray-tracing pipeline. *Right:* Final physically based renders.

Abstract. 3D Gaussian Splatting has recently revolutionised novel view synthesis as well as many other 3D vision methods and applications. Drawing inspiration from this representation, we now rethink textures to overcome the main issues of UV mapping while considerably lowering their memory footprint. Heat Kernel Textures (HKTex) eliminate UV unwrapping as well as their persistent issues of wasted UV space, seams, distortions, vertex-duplication, and varying resolution. Grounded in discrete Riemannian geometry and intrinsically defined on any manifold surface discretised as a triangular mesh, HKTex uses anisotropic heat kernels as geodesic equivalents to Gaussians. Like our kernels, also the optimisation of their position and the adaptive densification strategies were redefined to operate on the surface of the object to be textureised. Our novel representation is also fully integrated with a physically based renderer and can be optimised either from existing textures or multi-view images. Our project page and code are available at circle-group.github.io/research/HeatKernelTextures.

Keywords: Gaussian Splatting · Heat Kernel · Texture · Geodesic

1 Introduction

Rendering and representation techniques have evolved exponentially over the past decades, but the representation of objects' appearance is still mostly anchored in the traditional and fundamentally flawed UV-mapping. With this representation, points on the surface of a mesh are mapped onto image planes where

* Equal contribution

appearance properties are stored. Not only does this cause visual artifacts like seams, varying resolutions, and distortions, but it also incurs additional storage costs for saving the UV-coordinates associated with every vertex of a mesh and for duplicating vertices on seam lines. Empty UV-mapped regions further exacerbate memory waste. To overcome these limitations, we advocate for a new representation intrinsically defined on the surface of the objects and capable of lowering the memory footprint, with the potential to benefit games, virtual and augmented reality, and visual effects.

Multiple works have attempted to mitigate UV-related issues, most notably through implicit neural representations [19, 26] or coloured point clouds [12, 45, 46]. While promising, these approaches respectively incur considerable memory footprints, present topology-dependent structures, or require a substantial number of primitives. More recently, 3D Gaussian Splatting (3DGS) [18] has revolutionised novel view synthesis through a representation that tightly entangles geometry and appearance. However, standard 3DGS operates entirely in ambient Euclidean space. When applied to surface texturing, this assumption becomes a fundamental flaw: Euclidean optimisation naturally detaches Gaussians from the geometry, and the rigid 3D Gaussians themselves are incapable of bending to conform to the underlying manifold. While recent methods [8, 14] attempt to align Gaussians to the surface through the introduction of additional regularisation losses, the fundamental disconnect remains.

In contrast, we introduce a truly intrinsic, geodesic representation where the texturing primitives natively conform to the existing geometry of a mesh. Instead of operating in ambient 3D space, we replace Euclidean Gaussians with anisotropic heat kernels, defining all operations directly on the manifold surface. Specifically, kernel positions are updated through Riemannian geodesic optimisation, and our adaptive density controls also respect the manifold using the exponential map during kernel splits. To continuously and efficiently evaluate kernels across the surface, we devise new techniques to parametrise kernel anisotropies while simultaneously bridging continuous and discrete heat diffusion formulations. Our formulation enables seamless integration into physically based rendering pipelines where our textures can be differentiably ray-traced. It is important to note that our heat kernels are not splatted, but rather dynamically evaluated at ray intersections occurring during rendering. Without loss of generality, we focus this study on albedo texture properties, thus modelling how light reflects. We prove that Heat Kernel Textures (HKTex) can compress existing UV textures and also be used for inverse rendering from 2D photos, thus enabling distortion- and seam-free texture acquisition by construction.

To summarise, our main contributions are:

1. A **novel intrinsic texture representation** grounded in discrete Riemannian geometry that efficiently represents textures without UV-mapping.
2. A **Riemannian optimisation framework** that strictly constrains kernel movements and momentum to the surface of the mesh.
3. **Manifold-aware density controllers** specifically designed to operate alongside physically based rendering and our continuous texture evaluation.

4. The **integration of HKTex into a differentiable PBR** pipeline, enabling distortion-free texture optimisation from multi-view images.

2 Related Work

Neural Texture Representations. To circumvent the persistent artifacts of traditional UV-mapping, substantial effort has been directed towards coordinate-based neural networks to represent surface appearance. Early implicit approaches evaluated Multilayer Perceptrons (MLPs) directly on 3D Euclidean coordinates [26]. To improve spatial expressiveness, subsequent methods introduced hierarchical or multi-resolution per-vertex feature encodings that are interpolated and fed into MLPs [21]. Other hybrid approaches, such as NeuTex [42], attempted to map 3D points to an unfolded 2D space before neural evaluation. While these methods reduce reliance on manual UV-unwrapping, they remain fundamentally anchored in ambient 3D space or intermediate 2D mappings. Closely related to our mathematical foundation are Intrinsic Neural Fields [19], which bypass Euclidean coordinates by barycentrically interpolating the eigenvectors of the Laplace-Beltrami Operator (LBO) to condition an MLP. While mathematically elegant, MLP-based evaluations remain computationally heavy and lack the local, adaptive expressivity of explicit primitives.

Gaussian Splatting and Explicit Surfaces. Point-based graphics and surface splatting have long been utilized for rendering [51], but they have seen a massive resurgence following the introduction of 3D Gaussian Splatting (3DGS) [18]. The unprecedented speed and visual fidelity of 3DGS have inspired attempts to adapt Gaussians to 2D domains, such as replacing image pixels with explicit 2D Gaussians [48–50]. To apply this expressive power to 3D meshes, recent works have attempted to bridge GS with texture mapping. However, these methods typically regress to utilizing UV-parameterizations, either by storing Gaussians within the UV-images of a sphere [33] or by splatting within unfolded texture spaces [43]. Other advanced hybrid representations, such as Neural Shell Textures [47], use 2D Gaussians to represent local geometry while storing colors in multi-resolution hash grids [24]. Crucially, all of these GS-based approaches either rely on explicit UV-maps or operate in ambient Euclidean space, meaning their primitives cannot intrinsically bend or conform to general topological manifolds.

Integration with Physically Based Rendering. While 3DGS was designed for rapid rasterisation, its integration into ray-traced physically based rendering (PBR) is an emerging frontier. Recent works have incorporated Gaussians into PBR frameworks for phenomena like particle rendering [10], and extended them for complex inverse rendering and relighting tasks [13, 44]. However, these approaches still rely on volumetric clouds of Euclidean primitives. Our work aligns with this trajectory but fundamentally shifts the paradigm: rather than rasterising ambient Gaussians, we natively integrate intrinsic, manifold-bound heat kernels as a continuous, differentiable texture evaluated directly at ray intersections within a PBR engine.

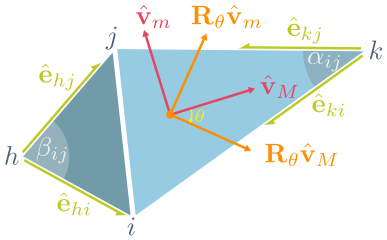
3 Background & Notation

Let $\mathcal{M} = \{\mathbf{X}, \mathbf{F}\}$ be a manifold mesh with $\mathbf{X} \in \mathbb{R}^{N\times 3} = [\mathbf{x}_1 \dots \mathbf{x}_N]^\top$ representing the positions of the N vertices used to discretise the continuous surface of an object, and $\mathbf{F} \in \mathbb{N}^{F\times 3}$ the face matrix indexing how vertices are connected into faces. We also group edges in a matrix $\mathbf{E} \in \mathbb{N}^{E\times 2}$. Points on the surface of $\mathcal{M}$ are defined by the pair $\boldsymbol{p} = (\mathrm{f}, \mathbf{b})$, where $\mathrm{f} \in [1, F]$ index the face $\mathbf{f}_\mathrm{f} \in \mathbf{F}$ on which a point lies, and $\mathbf{b} = [\mathrm{b}_u, \mathrm{b}_v, \mathrm{b}_w]^T$ are its barycentric coordinates on $\mathbf{f}_\mathrm{f}$. The coordinates of $\boldsymbol{p}$ are barycentrically interpolated from the vertex coordinates $\mathbf{x}_{\mathrm{f},0}, \mathbf{x}_{\mathrm{f},1}, \mathbf{x}_{\mathrm{f},2}$ of $\mathbf{f}_\mathrm{f}$ as: $\mathbf{p} = b^{\boldsymbol{p}}(\mathbf{X}) = \mathrm{b}_u\mathbf{x}_{\mathrm{f},0} + \mathrm{b}_v\mathbf{x}_{\mathrm{f},1} + \mathrm{b}_w\mathbf{x}_{\mathrm{f},2}$. Similarly, barycentric interpolation can be used to interpolate any other attribute $\mathbf{A} \in \mathbb{R}^{N\times A}$ defined on the vertices of the mesh. The simultaneous barycentric interpolation over multiple points $\boldsymbol{P} = \{\boldsymbol{p}_j\}_{j=1}^J$, is hereafter indicated as: $b^{\boldsymbol{P}}(\mathbf{A}) = \mathbf{A}^{\boldsymbol{P}} \in \mathbb{R}^{P\times A}$. Therefore, $b^{\boldsymbol{P}}(\mathbf{X}) = \mathbf{X}^{\boldsymbol{P}}$, which we also represent as $\mathbf{P}$ for simplicity.

Laplace Beltrami Operators. We now briefly introduce the Laplace Beltrami operator (LBO) in both its isotropic and anisotropic [2, 22, 40] versions. We generally define the Laplacian as $\mathbf{L}(\eta, \theta) = -\mathbf{M}\mathbf{W}(\eta, \theta)$, where $\mathbf{M}$ is the diagonal *mass matrix*, whose diagonal entries are proportional to the total area of the faces surrounding each vertex [12, 35], and $\mathbf{W}(\eta, \theta)$ is the *stiffness matrix* composed of weights:

$$\mathrm{w}_{ij} = \begin{cases} -\frac{1}{2}\left[\frac{\langle \hat{\mathbf{e}}_{kj}, \hat{\mathbf{e}}_{ki}\rangle_{\mathbf{H}_{\eta,\theta}}}{\sin(\alpha_{ij})} + \frac{\langle \hat{\mathbf{e}}_{hj}, \hat{\mathbf{e}}_{hi}\rangle_{\mathbf{H}_{\eta,\theta}}}{\sin(\beta_{ij})}\right] & (i,j) \in \mathbf{E}, \\ -\sum_{k\neq i} \mathrm{w}_{ik} & i = j, \\ 0 & \text{else.} \end{cases} \tag{1}$$

The inner products between the edge versors are weighted by a *shear matrix* $\mathbf{H}_{\eta,\theta} = \mathbf{R}_\theta \mathbf{U}_{ijk} \mathbf{D}_\eta \mathbf{U}_{ijk}^T \mathbf{R}_\theta^\top$. $\mathbf{U}_{ijk} = [\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{n}}]$ is the matrix containing a local reference frame defined on a face, with $\hat{\mathbf{n}}$ being the normal to the surface and $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ being the orthogonal components defined on the face —a common choice involves using the principal curvatures and setting $\hat{\mathbf{u}} = \hat{\mathbf{v}}_M$ and $\hat{\mathbf{v}} = \hat{\mathbf{v}}_m$. $\mathbf{R}_\theta$ is the rotation matrix rotating the reference frame, and $\mathbf{D}_\eta = \mathrm{diag}(\frac{1}{1+\eta}, 1, 1)$ is a 3×3 diagonal matrix controlling the anisotropy. When $\eta = \theta = 0$ we have $\mathbf{H}_{\eta,\theta} = \mathbf{I}$, thus $\langle \hat{\mathbf{e}}_{kj}, \hat{\mathbf{e}}_{ki}\rangle_{\mathbf{H}_{\eta,\theta}} = \hat{\mathbf{e}}_{kj}^\top \mathbf{H}_{\eta,\theta} \hat{\mathbf{e}}_{ki} = \cos(\alpha_{ij})$ and $\langle \hat{\mathbf{e}}_{kj}, \hat{\mathbf{e}}_{ki}\rangle_{\mathbf{H}_{\eta,\theta}} = \cos(\beta_{ij})$. This reduces Eq. (1) to the traditional isotropic LBO formulation, which we indicate with $\mathbf{L} = \mathbf{L}(0, 0)$. For $\eta, \theta \neq 0$ we have the anisotropic LBO (ALBO), which we indicate as $\mathbf{L}_{\eta,\theta} = \mathbf{L}(\eta, \theta)$.



The generalised eigendecomposition of the LBO can be written as $\mathbf{W}\boldsymbol{\Phi} = \mathbf{M}\boldsymbol{\Phi}\boldsymbol{\Lambda}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{N\times K} = [\boldsymbol{\phi}_1 \dots \boldsymbol{\phi}_K]^\top$ represents the first K eigenvectors defined at the vertices of the mesh, and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_K)$ is the diagonal matrix containing the eigenvalues. The eigendecomposition of the ALBO follows the same procedure to determine the η- and θ-dependent $\boldsymbol{\Phi}_{\eta,\theta}$ and $\boldsymbol{\Lambda}_{\eta,\theta}$.

Heat Diffusion and Kernels. The heat diffusion process over a mesh is mathematically described as: $\mathbf{L}\ u(\mathbf{x}_i, \mathrm{t}) = \frac{\partial u(\mathbf{x}_i, \mathrm{t})}{\partial t}$, where the vertex-valued scalar function $u : \mathcal{M} \times \mathbb{R}^+ \to \mathbb{R}$ represents how an initial heat distribution over $\mathcal{M}$ evolves for times $\mathrm{t} > 0$. Indicating with $\boldsymbol{u}_\mathrm{t}$ the vector obtained by evaluating $u(\mathbf{x}_i, \mathrm{t})$ for all vertices, the solution of the heat equation has the form [9,12,34,37]:

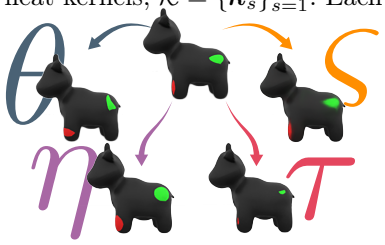
$$\boldsymbol{u}_\mathrm{t} = e^{-t\mathbf{L}}\boldsymbol{u}_0 \approx \boldsymbol{\Phi} e^{-t\boldsymbol{\Lambda}}\boldsymbol{\Phi}^T\mathbf{M}\boldsymbol{u}_0 \tag{2}$$

When the initial distribution is a delta function, $u(\mathbf{x}_i, 0) = \delta(\mathbf{x}_i) = \boldsymbol{\delta}_{\mathbf{x}_i}$, Eq. (2) is referred to as *heat kernel* and it represents how the heat generated by a single point source diffuses over the surface. In Euclidean space, it has the explicit expression of a Gaussian [36]. Note that while it is possible to simulate the diffusion by solving a sparse linear system, the spectral approach we adopt proved to be one order of magnitude faster and to scale to high-resolution meshes [34].

4 Heat Kernel Textures

We now detail our HKTex model, which uses arbitrarily positioned heat kernels to intrinsically represent textures directly on the surface of manifold meshes.

Kernel parameters. We define our HKTex as a collection of S optimisable heat kernels, $\mathcal{K} = \{\boldsymbol{\kappa}_s\}_{s=1}^{\mathrm{S}}$. Each kernel is parametrised as $\boldsymbol{\kappa} = \big(\boldsymbol{p}^\star, \theta, \eta, \tau, \varsigma, \mathbf{c}\big)$, where $\boldsymbol{p}^\star$ is the position of the heat source, θ and η are the angle and anisotropy controlling the diffusion process, τ is the threshold controlling the scale of the kernel, ς is the parameter controlling the sharpness of its border, and $\mathbf{c}$ is the RGB colour. The parameters of all $\mathcal{K}$ are jointly optimised.



Normalised heat kernels. In Sec. 3 we have introduced the discrete formulation of heat kernels and diffusion. While this is the gold standard on 3D meshes, we seek to break free from the mesh discretisation and represent textures beyond vertices. We thus leverage the continuous manifold formulation of heat kernels and define how to transform discrete quantities into continuous ones. Anisotropically heat diffusing a Dirac delta function centred in $\boldsymbol{p}^\star$ for a time t, we have the truncated continuous anisotropic heat kernel:

$$h_t\big(\boldsymbol{p}, \boldsymbol{p}^\star \mid \theta, \eta\big) = \sum_{k=1}^{K} e^{-t\lambda_k^{\eta,\theta}}\ \varphi_k^{\eta,\theta}(\boldsymbol{p})\ \varphi_k^{\eta,\theta}(\boldsymbol{p}^\star)\ m^\star, \tag{3}$$

where $\boldsymbol{p}$ is any arbitrary point on the surface of $\mathcal{M}$, $\lambda_k^{\eta,\theta}$ is the k-th eigenvalue with its corresponding eigenfunction $\varphi_k^{\eta,\theta}$ evaluated at $\boldsymbol{p}$ or $\boldsymbol{p}^\star$, and $m^\star$ is the mass at the source.

Since the physical heat diffusion process conserves energy, as heat spreads, the source temperature decreases. Therefore, the size and shape of kernels would affect their overall temperature. To stabilise optimisation and make filtering

predictable, we guarantee a maximum temperature of 1.0 at the kernel centre by normalising with respect to the post-diffusion source temperature:

$$\bar{h}_t\big(\boldsymbol{p} \,|\, \boldsymbol{p}^\star, \theta, \eta\big) = \frac{h_t\big(\boldsymbol{p}, \boldsymbol{p}^\star \,|\, \theta, \eta\big)}{h_t\big(\boldsymbol{p}^\star, \boldsymbol{p}^\star \,|\, \theta, \eta\big)}. \tag{4}$$

Biharmonic distance weighting. Heat kernels have a global support across the entire surface of the object. While they should decay with distance from the heat source, since we adopt a spectral truncation to the K most significant frequencies, the kernels suffer from a Gibbs-like phenomenon [30, 31] producing *spectral ringing* artifacts in distant regions (see supplementary materials). To suppress these unwanted artifacts and enforce local support, we introduce a Biharmonic distance Gaussian weighting that dampens the contributions of distant kernels. We compute the Biharmonic distance [20] using isotropic eigenproperties: $d_{\mathrm{BH}}(\boldsymbol{p}, \boldsymbol{p}^\star)^2 = \sum_{k=1}^{K} \frac{(\varphi_k^{\mathrm{iso}}(\boldsymbol{p}) - \varphi_k^{\mathrm{iso}}(\boldsymbol{p}^\star))^2}{(\lambda_k^{\mathrm{iso}})^2}$. Then, we weight our normalised kernels:

$$\tilde{h}_t(\boldsymbol{p} \,|\, \boldsymbol{p}^\star, \theta, \eta) = \bar{h}_t(\boldsymbol{p} \,|\, \boldsymbol{p}^\star, \theta, \eta) \cdot \exp\!\left(-\frac{d_{\mathrm{BH}}(\boldsymbol{p}, \boldsymbol{p}^\star)^2}{2\sigma_{\mathrm{BH}}^2}\right). \tag{5}$$

Kernel filtering. $\tilde{h}_t(\boldsymbol{p} \,|\, \boldsymbol{p}^\star, \theta, \eta)$ produces a Gaussian-like geodesic kernel, with values in $[0, 1]$, that gradually fades from the source location. Instead of optimising the diffusion time to scale the kernel, we fix t and modulate kernel scale (τ) and sharpness (ς) via a rescaled soft-step filtering function:

$$\alpha(\boldsymbol{p} \,|\, \boldsymbol{p}^\star, \theta, \eta, \tau, \varsigma) = \frac{\xi\big(\tilde{h}_t(\boldsymbol{p} \,|\, \boldsymbol{p}^\star, \theta, \eta)\big) - \xi(0)}{\xi(1) - \xi(0)}, \quad \text{with} \quad \xi(x) = \sigma\big(\varsigma \cdot (x - \tau)\big), \tag{6}$$

where $\sigma(\cdot)$ is a sigmoid function. This filtering preserves the $[0, 1]$ value range and provides more control over the kernel appearance, which can now be easily scaled down to point-sized dimensions while controlling the rate of signal decay. A filtered kernel can now either preserve its characteristic fading appearance or transform into a crisp geodesic ellipsoid defined on the surface of the mesh.

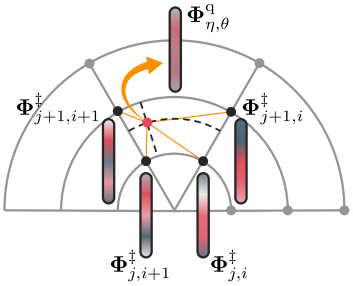
Colour formation. For a given kernel $\boldsymbol{\kappa}_s \in \mathcal{K}$, we denote its evaluated spatial footprint as $\alpha_s(\boldsymbol{p}) \equiv \alpha(\boldsymbol{p} \mid \boldsymbol{p}_s^\star, \theta_s, \eta_s, \tau_s, \varsigma_s)$. To determine the final texture colour $C(\boldsymbol{p})$ at a surface point $\boldsymbol{p}$, we need to aggregate the contributions of all kernels. Inspired by [49], we select the K_c top-contributing kernels to $\boldsymbol{p}$. In our formulation, this can be achieved by dynamically identifying the kernels with the highest $\alpha_s(\boldsymbol{p})$. The local texture is then computed via an alpha-weighted partition of unity over these selected kernels ($\mathcal{K}(\boldsymbol{p}) \subset \mathcal{K}$), which is added as a residual to an optimisable global base colour $\mathbf{c}_{\mathrm{base}}$:

$$C(\boldsymbol{p}) = \mathbf{c}_{\mathrm{base}} + \frac{\sum_{s \in \mathcal{K}(\boldsymbol{p})} \alpha_s(\boldsymbol{p})\, \mathbf{c}_s}{\sum_{s \in \mathcal{K}(\boldsymbol{p})} \alpha_s(\boldsymbol{p})}, \tag{7}$$

This colour formation paradigm guarantees a controlled blending between locally overlapping kernels, while empty regions seamlessly default to the underlying base colour.

4.1 Parametric Interpolation of the Spectral Bases

Interpolation of ALBO eigenproperties. Evaluating the continuous anisotropic heat kernel in Eq. (3) requires the spectral properties associated with a kernel's specific shape parameters (η, θ). Since constructing and eigendecomposing a unique ALBO for every kernel during an optimisation pipeline is computationally prohibitive, we devise a simplified and easily differentiable alternative. We thus discretise the parameter space of angles and anisotropies into a half-polar-like grid, where anisotropies are log-scaled to make the space more perceptually uniform and prevent high anisotropies from dominating. We associate each grid node (i, j), located at discrete angle θ_i and log-scaled anisotropy $s_j = \ln(1 + \eta_j)$, with its precomputed $\mathbf{\Phi}^{\ddagger}_{i,j}$ and $\mathbf{\Lambda}^{\ddagger}_{i,j}$. To evaluate a continuous kernel query with parameters (η, θ), we first project the queried anisotropy into the same logarithmic space, $s = \ln(1+\eta)$. We then perform a binary search to identify the bounding grid cell indices i and j such that $\theta_i \leq \theta < \theta_{i+1}$ and $s_j \leq s < s_{j+1}$. Within this cell, we compute the normalized local coordinates: $\Delta\theta = \frac{\theta-\theta_i}{\theta_{i+1}-\theta_i}$ and $\Delta s = \frac{s-s_j}{s_{j+1}-s_j}$. The queried eigenproperties are then efficiently approximated via bilinear interpolation over the four cell corners $\mathcal{C} = \{(a, b) \mid a \in \{j, j+1\}, b \in \{i, i+1\}\}$:



$$\mathbf{\Phi}^{\mathrm{q}}_{\eta,\theta} = \sum_{(a,b)\in\mathcal{C}} w_{a,b}\mathbf{\Phi}^{\ddagger}_{a,b} \qquad \text{and} \qquad \mathbf{\Lambda}^{\mathrm{q}}_{\eta,\theta} = \sum_{(a,b)\in\mathcal{C}} w_{a,b}\mathbf{\Lambda}^{\ddagger}_{a,b}, \tag{8}$$

where the interpolation weights $w_{a,b}$ are derived from the fractional distances $\Delta\theta$ and Δs (i.e., $w_{i,j} = (1 - \Delta\theta)(1 - \Delta s)$, $w_{i+1,j} = \Delta\theta(1 - \Delta s)$, etc.).

Alignment of precomputed ALBO eigenproperties. When eigendecomposing the precomputed ALBOs, we observe the same permutation, rotations, and sign ambiguities that are widely observed in shape matching problems (e.g., [27]). Since this behaviour would hinder the correctness of the bilinear interpolation in Eq. (8), we introduce an eigenproperties alignment procedure during the grid construction and thus ensure that all $\mathbf{\Phi}^{\ddagger}_{i,j}$ and $\mathbf{\Lambda}^{\ddagger}_{i,j}$ are consistent with the $\mathbf{\Phi}$ and $\mathbf{\Lambda}$ of the isotropic LBO. First, we resolve the permutation switching issue by maximizing the mass-weighted cross-correlation between two sets of eigenvectors: $\mathbf{\Phi}_{\mathrm{ref}}$ and $\mathbf{\Phi}^{\ddagger}_{i,j}$. We compute the correlation matrix $\mathbf{C} = \mathbf{\Phi}^{\top}_{\mathrm{ref}}\mathbf{M}\mathbf{\Phi}^{\ddagger}_{i,j}$ and apply the Hungarian algorithm to find the optimal assignment matrix $\mathbf{\Pi}$, yielding the permuted eigenvectors $\mathbf{\Phi}_{\mathrm{perm}} = \mathbf{\Phi}^{\ddagger}_{i,j}\mathbf{\Pi}$. Next, we solve the Orthogonal Procrustes problem minimising the mass-weighted Frobenius distance $\|\mathbf{\Phi}_{\mathrm{ref}} - \mathbf{\Phi}_{\mathrm{perm}}\mathbf{R}\|^2_{\mathbf{M}}$ via the Singular Value Decomposition (SVD) of $\mathbf{\Phi}^{\top}_{\mathrm{perm}}\mathbf{M}\mathbf{\Phi}_{\mathrm{ref}}$. The resulting rotation determines $\mathbf{\Phi}_{\mathrm{rot}} = \mathbf{\Phi}_{\mathrm{perm}}\mathbf{R}$. Finally, we correct the sign flips by constructing a diagonal matrix $\mathbf{S}$ where $\mathbf{S}_{k,k} = \mathrm{sgn}((\mathbf{\Phi}^{\top}_{\mathrm{ref}}\mathbf{M}\mathbf{\Phi}_{\mathrm{rot}})_{k,k})$. The fully aligned spectral basis stored at each grid node is thus obtained as $\mathbf{\Phi}_{\mathrm{aligned}} = \mathbf{\Phi}^{\ddagger}_{i,j}\mathbf{\Pi}\mathbf{R}\mathbf{S}$. Eigenvalues are simply reordered according to $\mathbf{\Pi}$. The initial reference eigenvectors are those of the LBO, then we use the last set

aligned during the grid construction. Hereafter, we consider all $\mathbf{\Phi}^{\ddagger}_{i,j}$ and $\mathbf{\Lambda}^{\ddagger}_{i,j}$ to be implicitly aligned for ease of notation.

4.2 Spatial Evaluation of Spectral Properties and Masses

Continuous spatial evaluation. All precomputed spectral properties are strictly defined on the vertices of $\mathcal{M}$. Limiting kernel placement and evaluation exclusively to these locations would bind our model to the surface discretisation. To overcome this limitation and evaluate any vertex-based spectral quantity $\mathbf{A}$ at an arbitrary continuous location $\boldsymbol{p}$, we exploit the inherent smoothness of the spectral bases. Inspired by [12], we approximate this continuous spatial evaluation via barycentric interpolation: $b^{\boldsymbol{p}}(\mathbf{A})$.

Biharmonic KNN search. Evaluating the heat equation for all S kernels and P target points using Eq. (3), yields an intractable $\mathcal{O}(P \times S)$ complexity. Because our biharmonic distance weighting (Eq. (5)) forces kernels to decay to zero outside a local radius, we can safely prune distant sources. We define a spectral embedding at the vertices as $\mathbf{Z} = \mathbf{\Phi}^{\text{iso}}(\mathbf{\Lambda}^{\text{iso}})^{-1}$. In this space, Euclidean distance corresponds to the biharmonic distance d_{BH} on the manifold. Using our continuous evaluation strategy, we compute these embeddings for kernel sources and target points as $\mathbf{z}(\boldsymbol{p}) \approx b^{\boldsymbol{p}}(\mathbf{Z})$ and $\mathbf{z}(\boldsymbol{p}^\star) \approx b^{\boldsymbol{p}^\star}(\mathbf{Z})$. We then construct a K-Nearest Neighbours (KNN) graph in this embedded space to dynamically restrict the evaluation of each point $\boldsymbol{p}$ to only its K_s nearest sources $\mathcal{N}(\boldsymbol{p})$.

Local anisotropic evaluation. Once the local subset of sources $\mathcal{N}(\boldsymbol{p})$ is identified, we must evaluate the continuous anisotropic eigenfunctions specifically required by those sources. Rather than computing the basis for all kernels globally, we retrieve the queried discrete eigenvectors $\mathbf{\Phi}^{\text{q}}_{\eta,\theta}$ corresponding to the specific shape parameters (θ_s, η_s) of only the K_s neighbouring sources. Applying our continuous evaluation principle, we approximate these source-specific eigenfunctions at both the source location $\boldsymbol{p}^\star$ and the target location $\boldsymbol{p}$. Specifically, we compute $[\varphi_1^{\eta,\theta}(\boldsymbol{p}^\star), \dots, \varphi_K^{\eta,\theta}(\boldsymbol{p}^\star)] \approx b^{\boldsymbol{p}^\star}(\mathbf{\Phi}^{\text{q}}_{\eta,\theta})$ and $[\varphi_1^{\eta,\theta}(\boldsymbol{p}), \dots, \varphi_K^{\eta,\theta}(\boldsymbol{p})] \approx b^{\boldsymbol{p}}(\mathbf{\Phi}^{\text{q}}_{\eta,\theta})$. This dual evaluation allows us to construct the heat kernel between $\boldsymbol{p}^\star$ and $\boldsymbol{p}$ (Eq. (3)), while keeping the computational complexity strictly bounded to $\mathcal{O}(P \times \mathrm{K}_s)$.

Mass definition for arbitrary points on $\mathcal{M}$. Finally, evaluating the continuous heat kernel (Eq. (3)) requires the mass $m^\star$ at the continuous source point $\boldsymbol{p}^\star$. Since the discrete mass matrix ($\mathbf{M}$) is proportional to vertex areas and thus local sampling density, we cannot simply use barycentric interpolations like we did for the eigenvectors.We thus formulate a Kernel Density Estimation (KDE) approach to correctly estimate the mass of every point based on the density of surrounding points (see supplementary materials). While this approach approximates the real mass, we empirically find that assigning a uniform unit mass to all sources ($m^\star = 1$) is significantly faster and does not deteriorate performance. Because the pointwise evaluation depends solely on the source mass as a linear scalar (Eq. (3)), during optimisation the other kernel properties can learn to compensate for the missing local integration weights.

4.3 Optimisation & Density Control

Geodesic position optimisation. Standard Euclidean optimisation would apply gradient updates directly to the 3D coordinates, inevitably pulling the heat kernel sources off the mesh surface. Since sources $\boldsymbol{p}^\star$ are intrinsically defined on $\mathcal{M}$, we must restrict movements exclusively to the surface. We thus formulate a Riemannian gradient descent optimiser: each gradient $\nabla_{\boldsymbol{p}^\star}\mathcal{L}$ is projected to the local tangent plane $\mathcal{T}_{\boldsymbol{p}^\star}\mathcal{M}$ and scaled to define an initial velocity $\mathbf{v}^\star$. We then compute the exponential map $\mathrm{Exp}(\boldsymbol{p}^\star, \mathbf{v}^\star)$, tracing the geodesic defined by this vector to determine the updated kernel position. To accelerate convergence with momentum, we parallel transport the accumulated momentum to the new kernel position. Both the exponential map and parallel transport are computed via the straightest geodesic algorithm [32], utilising the optimisation-friendly and CUDA-parallelised implementation introduced by [38].

Kernel pruning strategy. Like most Gaussian Splatting techniques, we adaptively control the density of our heat kernels throughout the optimisation process. In order to decrease computational overhead and storage costs, we aim to delete kernels that contribute little to the final texture. During an accumulation window Δ_{prune}, we record two metrics for every kernel $\boldsymbol{\kappa}_s \in \mathcal{K}$: a *hit count* $\nu(\boldsymbol{\kappa}_s)$ indicating how frequently it is selected in the local partition of unity $\mathcal{K}(\boldsymbol{p})$ (Eq. (7)), and an *energy score* $\omega(\boldsymbol{\kappa}_s)$ aggregating its filtered weights $\alpha_s(\boldsymbol{p})$. Periodically, we prune kernels that are either rarely hit ($\nu(\boldsymbol{\kappa}_s) < \gamma_{\text{hit}} \cdot \max(\{\nu(\boldsymbol{\kappa}_i)\}_{i=1}^{\text{S}})$) or energetically weak ($\omega(\boldsymbol{\kappa}_s) < \gamma_{\text{energy}}$).

Kernel densification strategy. To capture missing details, we dynamically densify kernels based on local reconstruction error. Over an accumulation window Δ_{error}, we compute the spatial error $e(\boldsymbol{p}) = \|\mathbf{c}(\boldsymbol{p}) - \mathbf{c}_{\text{GT}}(\boldsymbol{p})\|_1$ and distribute it to contributing kernels weighted by their normalised partition of unity, yielding a per-kernel score $E(\boldsymbol{\kappa}_s)$. Kernels exceeding an error threshold γ_{error} are densified according to their spatial extent $(1 - \tau_s)$. Small kernels $(1 - \tau_s \leq \gamma_{\text{size}})$ are *cloned* in place, allowing the optimiser to naturally separate them. Large kernels $(1 - \tau_s > \gamma_{\text{size}})$ are *split* into two smaller kernels. Crucially, to respect manifold constraints during splitting, we displace the new kernels along their principal axis using the exponential map provided by [38]. A maximum growth ratio γ_{growth} caps additions per step to ensure stability.

4.4 Rendering & Applications

Physically based rendering. Because our HKTex formulation is continuously evaluated across $\mathcal{M}$, it seamlessly integrates into modern physically based rendering (PBR) pipelines. We implement HKTex also as a differentiable texture within the `Mitsuba3` renderer [16]. During ray tracing, when a ray intersects the mesh at an arbitrary continuous point $\boldsymbol{p}$, the renderer queries the corresponding local subset of kernels $\mathcal{K}(\boldsymbol{p})$ and computes $C(\boldsymbol{p})$ following Eq. (7). By wrapping our `PyTorch`-based evaluation within `Dr.Jit` [15], we allow gradients

to backpropagate directly from the rendered image pixels to our intrinsic kernel parameters.

Optimise existing UV-textures. When a ground-truth UV texture is available, we can directly optimise our kernels. At each optimisation step, we uniformly sample a dense set of points $\boldsymbol{p}$ directly on the faces of $\mathcal{M}$, and evaluate their colour with $C(\boldsymbol{p})$. Our HKTex model is then updated by minimizing the photometric error with respect to $C_{\mathrm{GT}}(\boldsymbol{p})$, the colour queried from the UV texture at the same location: $\mathcal{L} = \|C(\boldsymbol{p}) - C_{\mathrm{GT}}(\boldsymbol{p})\|_2^2$.

Optimise multi-view images. Our HKTex model can also be optimised from multi-view images via the differentiable PBR renderer. In this scenario, we suppose the target mesh to be given alongside a set of camera poses. At each training step, we sample V camera views, identify all the rays hitting the mesh across views, and select a random subset to provide uniform coverage around the object. Minimising the rendering error against the ground-truth pixels at the ray intersections drives the HKTex optimisation.

5 Experiments

5.1 Data

We conduct experiments on a curated subset of Objaverse [11]. Since the full dataset comprises scraped online assets, mesh quality is not guaranteed. We therefore filter for objects that possess LVIS category labels, valid textures, and at least one community like and view, while excluding point clouds and animated models. To maintain a reasonable computational overhead, we restrict the maximum vertex count to $N = 60,000$. To ensure compatibility with our HKTex representation, we also define a rigorous geometric filtering: we exclude meshes with multiple disconnected components, with non-manifold geometries, and with complex internal structures (i.e. models presenting more than 20 intersections on any ray cast from a regular grid defined on the bounding box of the object). Finally, since our method is heavily reliant on the eigenfunctions of LBO, we reject models where such eigendecomposition fails. After filtering, we retain $\sim 3,000$ high-quality models. Note that while having an eigendecomposable LBO is essential, our filtering criteria can be relaxed at the cost of overall dataset quality or of separately processing parts of non-manifold and complex meshes. When fitting multi-view images, meshes are rendered with their albedos as the sole texture map.

5.2 Implementation details

Our HKTex pipeline is built with `PyTorch` [29], utilizing the GPU version of `FAISS` [17] for the biharmonic distance KNN search and `Mitsuba3` [16] with `Dr.Jit` [15] for differentiable ray tracing. Instead of building specialized CUDA kernels, we use `PyTorch`'s Triton code generation [3]. For all meshes, we precompute a 64-dimensional eigendecomposition of the isotropic LBO for calculating

biharmonic distances, and 256-dimensional eigendecompositions for all the ALBOs in our grid with the cross product of 7 angles and 7 anisotropies. The network parameters are initialized as in the following paragraph, and we use a projected gradient descent (PGD) algorithm to enforce the correct ranges. For all parameters except kernel locations, we use an Adam optimiser with a step-function learning rate scheduler for fitting UV-textures and multi-view images. To optimize kernel locations, we utilize a geodesic Riemannian-SGD algorithm on the GPU using the library from [38]. We run a short Bayesian hyperparameter tuning on 5 meshes excluded from the evaluation dataset for both setups using `Optuna` [1] with the Tree-Structured Parzen Estimator [4], and we share the exact hyperparameters in our supplementary. We use an importance sampling strategy, as detailed in our supplementary material, when fitting UV-textures. To render our output images and to perform differentiable ray tracing for inverse rendering, we utilize `Mitsuba3` [16] with `Dr.Jit` [15]. We use a path-replay backpropagation (PRB) integrator with a ray depth of 3 for training, and differentiate through the ray tracing process to optimize our textures. During inference, we only build the KNN database and pre-compute the self-reference heat kernels once, reducing computation significantly. We use the tuned hyperparameters to set the pruning and densification hyperparameters for both settings.

Initialisation. We use a uniform sampling strategy based on mesh area to first oversample a large number of points on the mesh surface, and then use farthest-point sampling to obtain kernel locations and face indices. The kernel colours are sampled from a uniform distribution between -1 and 1, with the mean subtracted and the mean set to our global base colour. The angles and anisotropies are sampled from a uniform distribution between 0 and π and 1 and 100, respectively. For sharpness and threshold initialization, we sample from the beta distributions $\mathcal{B}(2, 1)$ and $\mathcal{B}(10/7, 1)$, respectively. Then, the sharpnesses are scaled and shifted to the range 10-50, and the thresholds to 0.9-1.0 (0.7-1.0 for the multi-view rendering case). The parameters are then optimized, as detailed before and in the supplementary, using PGD.

5.3 Evaluation metrics

To quantitatively assess textures against the ground truth, we render five distinct views for each object and measure image metrics on them. In particular, we compute Mean Squared Errors (MSE) and Peak Signal-to-Noise Ratio (PSNR). To evaluate the preservation of structural details and contrast, we use the Structural Similarity Index (SSIM) and its multi-scale variant (MS-SSIM). Furthermore, we report the Learned Perceptual Image Patch Similarity (LPIPS) to measure perceptual quality. Finally, to validate our core claim of memory efficiency, we report the total Storage footprint in kilobytes (KB), strictly accounting for all intrinsic parameters, network weights, and any topological overheads (e.g., UV-coordinates) required by the baselines. We store with the same formats and without method-specific quantizations. Identical ZIP compression was already applied across all methods to provide a fair baseline for comparing intrinsic information density.

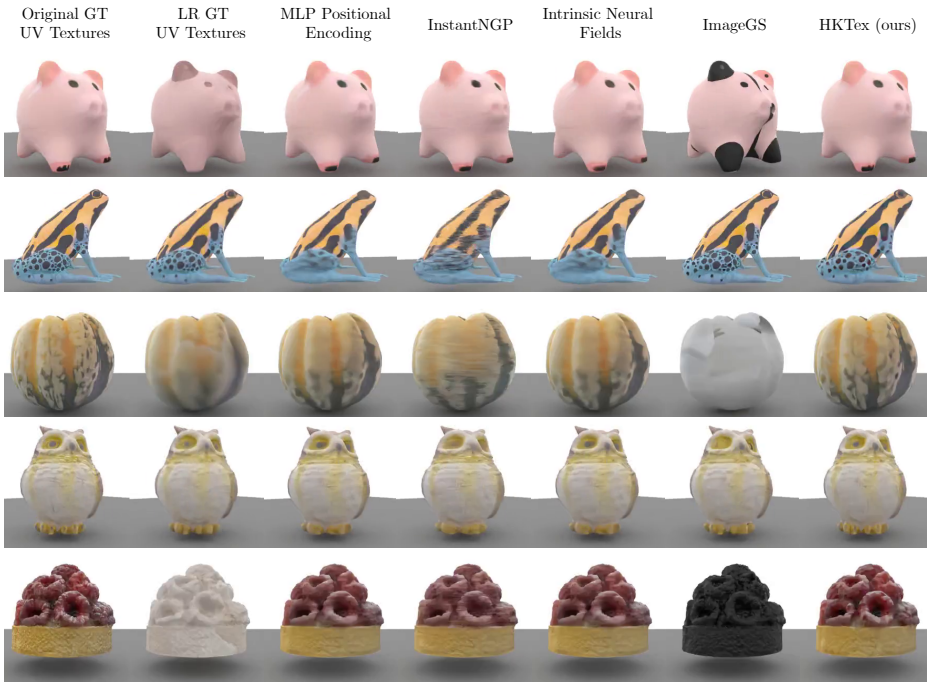


Fig. 2: Qualitative comparison against SotA methods. With the exception of the GT UVTextures, all other methods have similar or bigger storage size. For visual clarity, we render only the albedo textures under constant ambient illumination.

5.4 Fitting UV-textures

We first demonstrate our method's ability to represent existing UV textures at a significantly reduced storage cost. We optimise our HKTex against the ground truth albedo UV-texture of 313 meshes, arbitrarily selected from our curated dataset. We then quantitatively and qualitatively compare against leading state-of-the-art intrinsic texture representations and image-based baselines under strictly matched memory budgets.

For the neural baselines, we implement three multilayer perceptrons (MLPs) taking as input different positional encodings of the ray-intersection points. Specifically, we evaluate a standard MLP with traditional Fourier positional encoding [23], an adaptation of InstantNGP [24] utilizing hash-grid encoding, and Intrinsic Neural Fields [19], which barycentrically interpolates the eigenvectors of the LBO. To ensure a fair comparison, we conduct a grid search over the network width, depth, and hash-grid size to exactly match the storage cost of our optimised HKTex.

Furthermore, we compare against two explicit 2D baselines: downsampled ground-truth images and ImageGS [49] (a 2D Gaussian Splatting approach). Crucially, both image-based methods fundamentally rely on UV-mapping. Even

when the UVmap is provided, the storage of per-vertex UV-coordinates incurs a fixed memory overhead that must be accounted for in the total texture budget. Consequently, for meshes with high vertex counts, this overhead dominates the budget, severely restricting the memory available for the actual texture or 2D Gaussians. As shown in Tab. 1, an exact storage match is often impossible for these baselines; even when limiting the downsampled texture to a minimum resolution of 4×4 pixels and ImageGS to just 10 Gaussians, the performance penalty is catastrophic for densely sampled meshes (Fig. 2).

Fig. 3: Comparison with vertex colours.

Finally, we compare against vertex colours directly sampled from the GT UV-textures at the native mesh resolution (GT VTex) as well as iteratively subdividing the mesh to match our storage budget (HR GT VTex).

As reported in Tab. 1, HKTex achieves the lowest storage footprint after GT VTex while maintaining competitive reconstruction quality across all metrics. Although LR UV-textures achieve the highest PSNR due to directly storing image samples, our representation yields the second best perceptual quality (LPIPS) and the best SSIM, MS-SSIM, and MSE, indicating improved preservation of structural details despite using fewer parameters. Neural baselines and HR GT VTex require significantly larger storage budgets to achieve almost comparable accuracy, highlighting the efficiency of our geodesic heat kernel representations.

The qualitative comparisons in Fig. 2 further highlight the behaviour of the different representations. Neural field baselines tend to oversmooth fine texture details, particularly in regions with sharp colour transitions or high-frequency patterns (e.g., the frog and raspberry examples). ImageGS occasionally produces noticeable artifacts due to the limited number of splats available under the storage constraints, sometimes failing to capture the correct colour distribution. Vertex colours, even when subdivided (Fig. 3), fail to correctly represent textures because the vertex positioning is determined by geometry rather than texture content. In contrast, HKTex preserves sharper texture boundaries and more faithful colour distributions across objects while remaining within the same or smaller storage budgets.

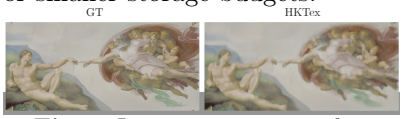


Fig. 4: Image on two triangles.

Fitting Images. To test HKTex's ability to represent high-resolution textures on coarse meshes, we fit an image onto a rectangle made of two triangles. In this special case of UV-texture fitting, we found it beneficial to amplify the weighted and normalised kernels as $(\tilde{h}_t)^a$ before filtering them (with $a = 30$). Nevertheless, $a = 1$ is sufficient for most meshes.

5.5 Fitting multi-view images

To demonstrate our method's ability to optimize multi-view images, we create a synthetic training scenario in which, instead of UV textures, we use rendered images of the ground-truth UV textures, following the same data setup in UV

Table 1: Quantitative comparison of the UV-texture fitting on 313 meshes from our curated subset of Objaverse [11]. HR and LR indicate high and low resolution.

	PSNR (↑)	LPIPS ($\times 10^{-2}$, ↓)	SSIM ($\times 10^{-2}$, ↑)	MS-SSIM ($\times 10^{-2}$, ↑)	MSE ($\times 10^{-3}$, ↓)	Render (ms, ↓)	Storage (KB, ↓)
LR GT UVTextures	**48.4** ±15.2	2.3 ±4.2	98.2 ±3.4	97.7 ±5.6	0.89 ±3.61	33.7 ±3.2	115.8 ±103.2
GT VTex (N≈7.7k)	37.9 ±8.5	4.0 ±5.0	96.8 ±4.2	96.1 ±6.6	0.94 ±2.33	**25.4** ±3.1	**17.3** ±32.6
HR GT VTex (N≈210k)	45.3 ±5.7	**1.1** ±1.7	98.8 ±2.1	99.4 ±1.4	**0.08** ±0.20	50.3 ±49.4	179.6 ±89.5
MLP Pos. Enc.	42.9 ±6.6	2.3 ±2.5	98.2 ±2.2	99.0 ±1.3	0.15 ±0.30	90.9 ±35.2	551.3 ±759.6
InstantNGP [24]	41.3 ±7.4	3.2 ±3.5	97.5 ±3.1	98.2 ±2.7	0.26 ±0.52	90.8 ±35.3	473.2 ±714.9
INFs [19]	42.5 ±6.8	2.7 ±3.4	98.0 ±2.7	98.8 ±1.9	0.19 ±0.44	91.4 ±35.3	928.2 ±1,439.8
ImageGS [49]	44.6 ±14.6	2.9 ±5.4	97.3 ±5.6	95.9 ±9.6	3.65 ±11.95	NA	115.4 ±93.9
HKTex (S ≈ 4.8k)	44.8 ±5.8	1.3 ±1.6	**98.9** ±1.5	**99.5** ±0.7	**0.08** ±0.20	784.7 ±474.3	96.6 ±12.2

texture fitting but with a smaller subset of the data. To generate training data, we sample rays from random views during training and select a random subset to form minibatches at each epoch.

Similar to the UV fitting setup, we implement two MLP architectures conditioned on different positional encodings, corresponding to the spatially varying albedo map. For a fair comparison, these representations are also trained using automatic differentiation and the interoperability between `Mitsuba3` and `PyTorch`. The first baseline utilizes traditional positional encodings. The second baseline is an adaptation of NVDiffRec [25], with the mesh and environment map fixed to the ground-truth mesh and environment map using hash encoding and with only the albedo map output. Like in the uv-fitting setting, we also conduct a grid search over the width, depth, and the hash-grid size to try to match the storage cost of our model.

As shown in Tab. 2, HKTex consistently outperforms the high-resolution vertex colours and neural baselines across all evaluation metrics while requiring substantially less storage. In particular, the improvements in LPIPS and SSIM indicate that our intrinsic formulation preserves perceptual details and structural consistency more effectively during inverse rendering. These results suggest that constraining the representation directly to the surface manifold provides a strong inductive bias for recovering appearance from sparse multi-view observations.

While our representation introduces additional evaluation overhead due to kernel aggregation and geodesic neighbourhood queries as seen in Table 1, this cost is partially amortized in the multi-view setting where ray tracing (with an increased depth) dominates the rendering time as seen in Table 2.

6 Conclusion

In this work, we presented HKTex, an intrinsic texture representation that replaces traditional UV-mapped textures with anisotropic heat kernels defined directly on the surface manifold. By operating entirely in the intrinsic geometry of the mesh, our formulation eliminates seams, distortions, and redundant storage associated with UV parametrisations. Through the use of geodesic heat

Table 2: Quantitative evaluation of the multi-view rendering on a curated subset of 162 meshes from Objaverse [11]. NVDiffRec [25] is our adaptation with the ground truth mesh and environment map fixed instead of learned, and with only albedo map output like our method.

	PSNR (↑)	LPIPS ($\times 10^{-2}$, ↓)	SSIM ($\times 10^{-2}$, ↑)	MS-SSIM ($\times 10^{-2}$, ↑)	MSE ($\times 10^{-3}$, ↓)	Render (s, ↓)	Storage (KB, ↓)
MLP Pos. Enc.	35.75 ±7.76	3.7 ±3.8	97.2 ±3.1	96.8 ±4.9	1.55 ±4.83	0.68 ±0.12	602.71 ±755.89
NVDiffRec* [25]	36.40 ±6.83	3.4 ±3.6	97.4 ±2.9	97.4 ±3.5	0.79 ±2.04	0.66 ±0.11	516.12 ±722.66
HR VTex	37.16 ±5.69	3.1 ±3.4	97.5 ±2.8	97.9 ±2.6	0.44 ±0.85	**0.58 ±0.08**	81.06 ±131.16
HKTex (S ≈ 3.8k)	**37.61 ±4.71**	**2.1 ±1.9**	**98.3 ±1.6**	**98.6 ±1.4**	**0.30 ±0.53**	1.2 ±0.3	**78.73 ±23.12**

kernels and Riemannian optimisation, our approach maintains strict alignment with the underlying surface while enabling continuous appearance evaluation within a differentiable physically based renderer. This formulation enables both the compression of existing UV textures and the direct reconstruction of surface appearance from multi-view imagery, while remaining entirely intrinsic to the mesh and independent of any global surface parametrisation. HKTex's mathematical continuity essentially acts as diffusion curves on surfaces, establishing a foundational step toward manifold vector graphics.

Limitations and future work. Future work may extend the representation beyond albedo to additional appearance attributes such as spatially varying BSDF parameters. While our current evaluation focuses on albedo textures, each kernel could be augmented with additional material parameters (e.g., roughness or specular components) and optimized through differentiable rendering with `Mitsuba3`.

Acknowledgements

T. Birdal was supported by a UKRI Future Leaders Fellowship [grant number MR/Y018818/1]. S. Foti, S. Zafeiriou, and T. Birdal were supported by the EPSRC Project GNOMON (EP/X011364/1). S. Foti and S. Zafeiriou were also supported by the Turing AI Fellowship MAGAL (EP/Z534699/1).

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework (2019), `https://arxiv.org/abs/1907.10902`
2. Andreux, M., Rodola, E., Aubry, M., Cremers, D.: Anisotropic laplace-beltrami operators for shape analysis. In: European conference on computer vision. pp. 299–312. Springer (2014)
3. Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B.,

Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C.K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M.Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., Chintala, S.: Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. p. 929–947. ASPLOS '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3620665.3640366, https://doi.org/10.1145/3620665.3640366

4. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 24. Curran Associates, Inc. (2011), https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
5. Burley, B.: Physically Based Shading at Disney. SIGGRAPH 2012 Course Notes (2012), https://www.disneyanimation.com/publications/physically-based-shading-at-disney/
6. Burley, B.: Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering. SIGGRAPH 2015 Course Notes (2015), https://blog.selfshadow.com/publications/s2015-shading-course/#course_content
7. Chaitanya, C.R.A., Kaplanyan, A.S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., Aila, T.: Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. ACM Trans. Graph. **36**(4) (Jul 2017). https://doi.org/10.1145/3072959.3073601, https://doi.org/10.1145/3072959.3073601
8. Choi, J., Lee, Y., Lee, H., Kwon, H., Manocha, D.: Meshgs: Adaptive mesh-aligned gaussian splatting for high-quality rendering. In: Proceedings of the Asian Conference on Computer Vision. pp. 3310–3326 (2024)
9. Chung, M.K., Qiu, A., Seo, S., Vorperian, H.K.: Unified heat kernel regression for diffusion, kernel smoothing and wavelets on manifolds and its application to mandible growth modeling in ct images. Medical image analysis **22**(1), 63–76 (2015)
10. Condor, J., Speierer, S., Bode, L., Bozic, A., Green, S., Didyk, P., Jarabo, A.: Don't splat your gaussians: Volumetric ray-traced primitives for modeling and rendering scattering and emissive media. ACM Transactions on Graphics **44**(1), 1–17 (2025)
11. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13142–13153 (2023)
12. Foti, S., Zafeiriou, S., Birdal, T.: Uv-free texture generation with denoising and geodesic heat diffusion. Advances in Neural Information Processing Systems **37**, 128053–128081 (2024)
13. Gao, J., Gu, C., Lin, Y., Li, Z., Zhu, H., Cao, X., Zhang, L., Yao, Y.: Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing. In: European Conference on Computer Vision. pp. 73–89. Springer (2024)
14. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5354–5363 (2024)

15. Jakob, W., Speierer, S., Roussel, N., Vicini, D.: Dr.jit: A just-in-time compiler for differentiable rendering. Transactions on Graphics (Proceedings of SIGGRAPH) **41**(4) (Jul 2022). https://doi.org/10.1145/3528223.3530099
16. Jakob, W., Speierer, S., Roussel, N., Nimier-David, M., Vicini, D., Zeltner, T., Nicolet, B., Crespo, M., Leroy, V., Zhang, Z.: Mitsuba 3 renderer (2022), https://mitsuba-renderer.org
17. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2019)
18. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
19. Koestler, L., Grittner, D., Moeller, M., Cremers, D., Lähner, Z.: Intrinsic neural fields: Learning functions on manifolds. In: European Conference on Computer Vision. pp. 622–639. Springer (2022)
20. Lipman, Y., Rustamov, R.M., Funkhouser, T.A.: Biharmonic distance. ACM Transactions on Graphics (TOG) **29**(3), 1–11 (2010)
21. Mahajan, M., Hofherr, F., Cremers, D.: Meshfeat: Multi-resolution features for neural fields on meshes. In: European Conference on Computer Vision. pp. 268–285. Springer (2024)
22. Melzi, S., Rodolà, E., Castellani, U., Bronstein, M.M.: Shape analysis with anisotropic windowed fourier transform. In: 2016 Fourth International Conference on 3D Vision (3DV). pp. 470–478. IEEE (2016)
23. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
24. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM transactions on graphics (TOG) **41**(4), 1–15 (2022)
25. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting Triangular 3D Models, Materials, and Lighting From Images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8280–8290 (June 2022)
26. Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., Geiger, A.: Texture fields: Learning texture representations in function space. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4531–4540 (2019)
27. Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: a flexible representation of maps between shapes. ACM Transactions on Graphics (ToG) **31**(4), 1–11 (2012)
28. Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: Optix: a general purpose ray tracing engine. ACM Trans. Graph. **29**(4) (Jul 2010). https://doi.org/10.1145/1778765.1778803, https://doi.org/10.1145/1778765.1778803
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library (2019), https://arxiv.org/abs/1912.01703
30. Patané, G.: Star-laplacian spectral kernels and distances for geometry processing and shape analysis. In: Computer Graphics Forum. vol. 35, pp. 599–624. Wiley Online Library (2016)
31. Patané, G.: Spectral laplace transform of signals on arbitrary domains. Journal of Scientific Computing **96**(3), 65 (2023)

32. Polthier, K., Schmies, M.: Straightest geodesics on polyhedral surfaces. In: ACM SIGGRAPH 2006 Courses. p. 30–38. SIGGRAPH '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1185657.1185664, https://doi.org/10.1145/1185657.1185664
33. Rai, A., Wang, D., Jain, M., Sarafianos, N., Chen, K., Sridhar, S., Prakash, A.: Uvgs: Reimagining unstructured 3d gaussian splatting using uv mapping. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5927–5937 (2025)
34. Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M.: Diffusionnet: Discretization agnostic learning on surfaces. ACM Transactions on Graphics (TOG) **41**(3), 1–16 (2022)
35. Sharp, N., Crane, K.: A laplacian for nonmanifold triangle meshes. In: Computer Graphics Forum. vol. 39, pp. 69–80. Wiley Online Library (2020)
36. Sharp, N., Soliman, Y., Crane, K.: The vector heat method. ACM Transactions on Graphics (TOG) **38**(3), 1–19 (2019)
37. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Computer graphics forum. vol. 28, pp. 1383–1392. Wiley Online Library (2009)
38. Verninas, H., Korkmaz, C., Zafeiriou, S., Birdal, T., Foti, S.: Parallelised differentiable straightest geodesics for 3d meshes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2026)
39. Vicini, D., Speierer, S., Jakob, W.: Path replay backpropagation: Differentiating light paths using constant memory and linear time. Transactions on Graphics (Proceedings of SIGGRAPH) **40**(4), 108:1–108:14 (Aug 2021). https://doi.org/10.1145/3450626.3459804
40. Wardetzky, M., Mathur, S., Kälberer, F., Grinspun, E.: Discrete laplace operators: no free lunch. In: Symposium on Geometry processing. vol. 33, p. 37. Aire-la-Ville, Switzerland (2007)
41. Wiersma, R., Philip, J., Hasan, M., Mullia, K., Luan, F., Eisemann, E., Deschaintre, V.: Uncertainty for svbrdf acquisition using frequency analysis. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–12 (2025)
42. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7119–7128 (2021)
43. Xu, T.X., Hu, W., Lai, Y.K., Shan, Y., Zhang, S.H.: Texture-gs: Disentangling the geometry and texture for 3d gaussian splatting editing. In: European Conference on Computer Vision. pp. 37–53. Springer (2024)
44. Ye, K., Gao, C., Li, G., Chen, W., Chen, B.: Geosplatting: Towards geometry guided gaussian splatting for physically-based inverse rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 28991–29000 (2025)
45. Yuksel, C.: Mesh color textures. In: Proceedings of High Performance Graphics. HPG '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3105762.3105780, https://doi.org/10.1145/3105762.3105780
46. Yuksel, C., Keyser, J., House, D.H.: Mesh colors. ACM Transactions on Graphics (TOG) **29**(2), 1–11 (2010)

47. Zhang, X., Chen, A., Xiong, J., Dai, P., Shen, Y., Xu, W.: Neural shell texture splatting: More details and fewer primitives. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 25229–25238 (2025)
48. Zhang, X., Ge, X., Xu, T., He, D., Wang, Y., Qin, H., Lu, G., Geng, J., Zhang, J.: Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In: European Conference on Computer Vision. pp. 327–345. Springer (2024)
49. Zhang, Y., Li, B., Kuznetsov, A., Jindal, A., Diolatzis, S., Chen, K., Sochenov, A., Kaplanyan, A., Sun, Q.: Image-gs: Content-adaptive image representation via 2d gaussians. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. pp. 1–11 (2025)
50. Zhu, L., Lin, G., Chen, J., Zhang, X., Jin, Z., Wang, Z., Yu, L.: Large images are gaussians: High-quality large image representation with levels of 2d gaussian splatting. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 10977–10985 (2025)
51. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 371–378 (2001)