EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

Implicit UVs: Real-time semi-global parameterization of implicit surfaces - Supplemental Material

Baptiste Genest[1], Pierre Gueth[2], Jérémy Levallois[2] and Stephanie Wang[2]

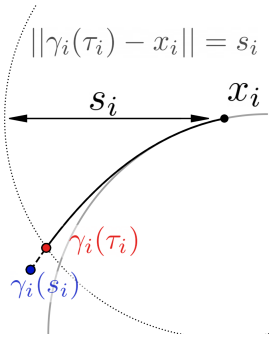
[1]Univ Lyon, CNRS, Lyon1, INSA, LIRIS, France
[2]Adobe Research

A Adaptive step size

One key ingredient in ensuring the continuity of the local parameterization is requiring that the algorithm takes the same number of steps at all query points. Take the case that x and x' are two nearby query points. Because we run Alg. **??** on x and x' independently, if one uses N steps and the other uses $N-1$ steps, that missing step can introduce discontinuities in their uv-coordinates. Using a fixed number of iterations with adaptive step size also helps evenly allocate computation time for each core of the parallel computation.

To fully utilize every iteration of the N steps, we must pick the step size τ_i of the gradient descent to maximize our chances of obtaining an almost-geodesic curve γ that reaches the reference point p. If the step size is too large, we can deviate from the level set M if it is highly curved; if it is too small, we might not reach the seed before the iterations run out. We set the time steps so that each iteration moves by the Euclidean distance between the current location and the goal $\|x_i - p\|$, divided by the remaining number of steps:

$$s_i := \|x_{i+1} - x_i\| = \frac{\|x_i - p\|}{N - i}. \quad (1)$$



To infer the step size τ_i required for the above equation, we take from Eq. **??** that $x_{i+1} = \gamma_i(\tau_i) = x_i + \tau_i v + \frac{\tau_i^2}{2}\alpha n$ for some scalar $\alpha \in \mathbb{R}$ and $||v|| = ||n|| = 1, v \perp n$. We then solve $\|\gamma_i(\tau_i) - x_i\|^2 = s_i^2$ for τ_i:

$$\tau_i = \sqrt{\frac{\sqrt{1 + s_i^2\alpha^2} - 1}{\frac{\alpha^2}{2}}} \leq s_i. \quad (2)$$

Note when the surface is flat, $H_f = 0$ and therefore $\alpha = 0$, we recover $\tau_i = s_i$. When the surface M is smooth, the fact that each query point takes the same number of steps with position-dependent step size makes the resulting log map depend continuously on the input. When the surface is merely C^1 and not C^2 (differentiable but not twice-differentiable), we found that reprojecting each x_i using Eq. **??** is enough in practice.

B Approximation of the geodesic distance to the Voronoï frontier

In a purely Euclidean setting, one can evaluate the signed distance from a query point x to the Voronoï frontier $V_{i,j}$ between two seeds p_i and p_j by projecting the vector $x - p_i$ to the line connecting p_i and p_j:

$$d^{\text{Euc}}_{V_{i,j}}(x) = \left\langle x - p_i, \frac{p_j - p_i}{||p_j - p_i||} \right\rangle - \frac{||p_j - p_i||}{2}. \quad (3)$$

To replace the inner product structure with only length information, we first replace $\langle x - p_i, p_j - p_i \rangle$ with $||p - p_i||||p_j - p_i|| \cos(\alpha)$ and use the cosine law,

$$\cos(\alpha) = \frac{l_0^2 + l_1^2 - l_2^2}{2 l_0 l_1}, \quad (4)$$

where $l_0 = d(x, p_i), l_1 = d(p_j, p_i), l_2 = d(x, p_j)$. Combining these formulas gives:

$$\begin{aligned} d^{\text{Euc}}_{V_{ij}}(x) &= \frac{||x - p_i||^2 + ||p_j - p_i||^2 - ||x - p_j||^2}{2||p_j - p_i||} - \frac{||p_j - p_i||}{2} \\ &= \frac{||x - p_i||^2 - ||x - p_j||^2}{2||p_j - p_i||}. \end{aligned} \quad (5)$$

To get an approximation of the *geodesic* distance to the geodesic Voronoï frontier, we replace all the Euclidean distances with the geodesic ones:

$$d_{V_{ij}}(x) \approx \frac{d_M(x, p_i)^2 - d_M(x, p_j)^2}{2 d_M(p_j, p_i)}. \quad (6)$$

C Detecting topological obstruction

By definition, there exists no continuous and global mapping between geometries of different topologies. For example, we cannot map a cylinder to a plane without cutting the cylinder first. This is why placing three seeds on a cylinder and merging their uv-fields would not yield a valid global mapping. A seam must exist between at least two seeds. We want to detect such topological obstructions when the user inputs the merging graph so we can alert them that they are about to create an incoherent uv-field (See Fig. 1). Even

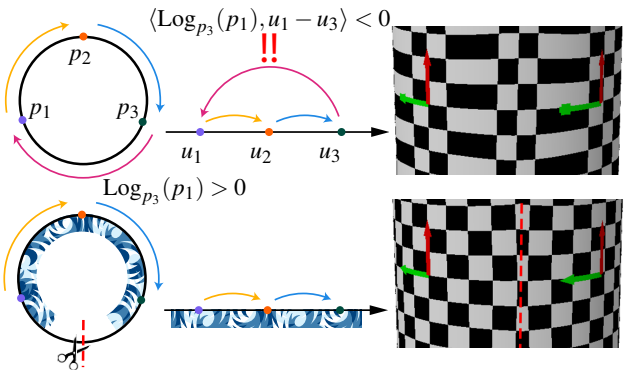


Figure 1: *In 1D, a circle cannot be parametrized by a straight line without inversion (top left, top middle). This is why a cylinder cannot be parametrized by a flat plane of texture (top right) without texture inversion. We detect this inversion between seeds and suggest a cut at their Voronoï frontier (bottom row).*

though we avoid global operations to ensure a real-time pipeline, we can reliably detect uv defects caused by topology by checking if the global uv (Eq. **??**) has a negative Jacobian, i.e. $\det(J) < 0$. It turns out we only need to check for

$$\left\langle \mathrm{Log}_{p_i}(p_j), u_j - u_i \right\rangle < 0, \tag{7}$$

for each blending edge $E_{ij} \in E$ after computing the uv-offsets.

D Interpolation and diffusion of scalar and vector quantities

One can use our fast estimate of geodesics and parallel transport to diffuse scalar or vector quantities. This can be used for painting / designing textures, for example.

Diffusion

Diffusion of quantities $\{a_i\}_{i=1}^N$ from N point sources $\{p_i\}_{i=1}^N$ is often solved with the *heat equation*:

$$\begin{cases} \partial_t u(x,t) = \Delta u(x,t), & t > 0 \\ u(x,0) = g(x), & t = 0, \end{cases} \tag{8}$$

where the initial value is the *Dirac-delta* of the quantities to be diffused,

$$g(x) = \sum_{i=1}^{N} a_i \delta_{p_i}(x). \tag{9}$$

The usual way to solve this problem on a meshed surface is to discretize the Laplacian Δ (e.g. using the cotan Laplacian) and then perform a time integration. Of course, here we want to avoid meshing the surface and solving the global problem.

We can mimic the solution to the heat equation by exploiting the geodesic distance between the source points and a query point $x \in M$. The fundamental solution to the heat equation is the *heat kernel*,

$$K(x,t) = \frac{1}{4\pi t} \exp\left(\frac{-||x||^2}{4t}\right). \tag{10}$$

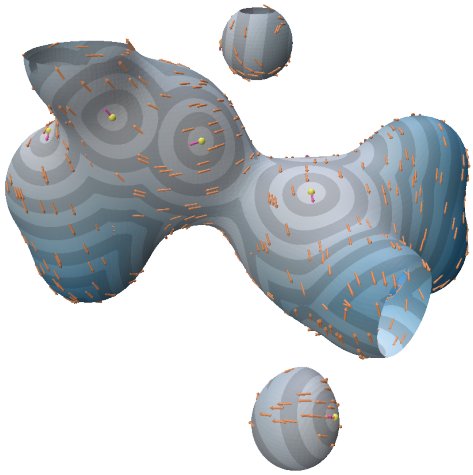


Figure 2: *Tangent vector field interpolation on a metaball surface by kernel methods using the approximated geodesic distance.*

By linearity, the solution to Eq. 8 is then $u(x,t) = \sum_i K(x - p_i, t) a_i$. We substitute $\|\cdot\|$ in the heat kernel with the geodesic distance and approximate diffusion on a curved surface M by

$$u(x,t) \approx \sum_{i=1}^{N} \frac{1}{4\pi t} \exp\left(\frac{-d_M(x,p_i)^2}{4t}\right) a_i \tag{11}$$

Interpolation

In a similar fashion, we can use kernel methods to interpolate values instead of diffusing them. For example, one can use a Gaussian kernel,

$$u_\sigma(x) = \frac{\sum_{i=1}^{N} \exp\left(-d_M(x,p_i)^2 \sigma^{-1}\right) a_i}{\sum_{i=1}^{N} \exp\left(-d_M(x,p_i)^2 \sigma^{-1}\right)}, \tag{12}$$

to interpolate the data $\{a_i\}$.

Diffusion on tangent spaces

We can use Eq. 11 to diffuse $\mathbb{R}^3$-vectors, but the result wouldn't be a field of tangent vectors even if the inputs are $(v_i \in \mathrm{T}_{p_i}\mathrm{M}, i = 1, \cdots, N)$. Therefore, one must use parallel transport to first bring all the vectors $v_i \in \mathrm{T}_{p_i}\mathrm{M}$ to the tangent space $\mathrm{T}_x\mathrm{M}$ at the query point x. We use the R matrices built by Alg. **??** for our point-wise interpolator:

$$v_\sigma(x) = \frac{\sum_{i=1}^{N} \exp\left(-d_M(x,p_i)^2 \sigma^{-1}\right) R_{p_i \to x}(v_i)}{\sum_{i=1}^{N} \exp\left(-d_M(x,p_i)^2 \sigma^{-1}\right)}. \tag{13}$$

The result of this interpolator is displayed in Fig. 2.

E Approximating normals from displacement

When no normal map is precomputed, we can still avoid using finite difference schemes on f_h to compute the normal of the displaced SDF.

We start from the fact that the vector $\nabla f(x)$ can be represented from any ortho-normal basis $\{e'_1(x), e'_2(x), e'_3(x)\}$:

$$\nabla f(x) = \sum_{i=1}^{3} D_{e'_i} f(x) e'_i(x). \tag{14}$$

Similar to Sec. **??**, we use the parallel-transported frame $e'_1 = R_{p_i \to x} e_1, e'_2 = R_{p_i \to x} e_2$ from the nearest seed $(p_i, (e_1, e_2)_i)$ and the normal $e'_3 = n_f(x)$ for this ortho-normal basis. We know $D_{n(x)} f_h(x) = 1$ because Eq. **??** and $D_n \Pi_f = 0$ and $D_n f(x) = 1$ as f is an SDF. The true derivatives w.r.t. $e'_1(x), e'_2(x)$ need to consider the change of the frame $e'_1(x), e'_2(x)$, but we ignore it by assuming that the variations from the height map h dominates the derivatives. We thus obtain, for $\xi \in f_h^{-1}(0)$ on the height-displaced surface and $x = \Pi_f(\xi) \in f^{-1}(0)$,

$$\nabla f_h(\xi) \approx \nabla f_h(x) \approx \nabla f(x) + e'_1 \partial_1 h(u(x)) + e'_2 \partial_2 h(u(x)). \tag{15}$$

We approximate the partial derivatives $\partial_1 h, \partial_2 h$ using finite difference schemes on $\mathbb{R}^2$. Finally, we normalize ∇f_h to obtain the unit normal vector.

It's worth noting that while this approximation is acceptable for visualization, it cannot provide reliable bounds for the Lipschitz constant $\max_{\xi \in \mathbb{R}^3} \|\nabla f_h(\xi)\|$ of the displaced SDF f_h. Computing the exact Lipschitz bound would involve knowing the curvature of the surface and parameterization distortion *a priori*. We cannot derive these pieces of information from our point-wise and seed-placement-dependent parameterization easily. However, we found that using a multiple of the standard Lipschitz constant used for height maps [GGP*15]

$$\lambda_h = \max_{x \in \mathbb{R}^2} \sqrt{1 + ||\nabla h(x)||^2}, \tag{16}$$

is usually enough in practice.

References

[GGP*15] GÉNEVAUX J.-D., GALIN E., PEYTAVIE A., GUÉRIN E., BRIQUET C., GROSBELLET F., BENES B.: Terrain modelling from feature primitives. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 198–210. 3