Decorating 3D Models with Poisson Vector Graphics

Qian Fu[a], Fei Hou[b], Qian Sun[c], Shiqing Xin[d], Yongjin Liu[e], Wencheng Wang[b], Hong Qin[f], Ying He[a,*]

[a]*Nanyang Technological University, Singapore*
[b]*State Key Laboratory of Computer Science, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China*
[c]*Tianjin University, China*
[d]*Shandong University, China*
[e]*Tsinghua University, China*
[f]*Stony Brook University, United States*

Abstract

This paper proposes a novel method for decorating 3D surfaces using a new type of vector graphics, called Poisson vector graphics (PVG). Unlike other existing techniques that frequently require local/global parameterization, our approach advocates a parameterization-free paradigm, affording decoration of geometric models with any topological type while minimizing the overall computational expenses. Since Poisson Vector Graphics supports a set of simple discrete curves, it is straightforward for users to edit colors and synthesize geometry details. Meanwhile, the details could be organized by Poisson Region (PR), leading to much smoother decoration than those of Diffusion Curve (DC). Consequently, it is an ideal tool to create smooth relief. It may be noted that, DC is adequate to create sharp or discontinuous results. But PR is superior to DC, supporting level-of-details editing on meshes thanks to its smoothness. To render PVG on meshes efficiently, we develop a Poisson solver based on harmonic B-splines, which could be constructed using geodesic Voronoi diagram. Our Poisson solver is a local solver for rendering with more flexibility and versatility. We demonstrate the efficacy of our approach on synthetic and real-world 3D models.

Keywords: Poisson vector graphics, Poisson solver, harmonic B-splines function, diffusion curves, Poisson regions, geometric textures

1. Introduction and Motivation

Despite many novel techniques being developed for 3D graphics modeling in recent decades, decorating 3D surfaces of arbitrary topological type still remains a common yet challenging task in geometric modeling and computer graphics applications. Texture mapping is still a popular technique for enhancing the realism of 3D surfaces, and the other commonly-used technique is bump mapping. Based on the local details of the texture, Sander et al. [1] reallocated the object parametrization to locally provide more details on the mesh. Carr and Hart [2] used the surface painting to paint a texture directly onto a surface with a texture atlas. To provide a more clear shadow map result near the camera according to the current viewpoint, Stamminger and Drettakis [3] changed the texture space by perspective shadow map. Dachsbacher and Stamminger [4] rendered procedural terrain via geometry image warping. However, since local or global parameterization is required, texture mapping is oftentimes computationally expensive. From the authoring perspective, 3D editing tools could not be easily supported by texture mapping, as texture mapping frequently fails to handle models with complex geometry, and it also tends to produce high distortion.

An alternative approach is to directly construct color and displacement functions on 3D surfaces using vector graphics. Orzan et al. [5] proposed diffusion curves, which propagate colors smoothly from curves, producing smooth color distribution in space except at the curves. Based on diffusion curves, Jeschke et al. [6] proposed to render surface details. And then they extended diffusion curves with view dependent and dynamic feature embedding to render details on 3D meshes. Since the solution is C^0 continuity on diffusion curves, they cannot produce smooth details.

Strongly inspired by the existing works on decoration of arbitrary mesh, we take a different approach and propose a 3D mesh decoration algorithm based on Poisson vector graphics in order to produce smooth details. PVG [7] is a natural extension to the popular diffusion curves with two new primitives — Poisson curve (PC) and Poisson region, which is a good 2D graphic authoring tool based on Poisson's equation. It extends diffusion curve from Laplacian's equations to Poisson's equation, allowing non-zero Laplacians away from curves. In this paper, we generalize PVG from planar domain to 3D mesh decoration. Diffusion curves based decoration algorithm suffers from C^0 or C^{-1} continuity. In contrast, thanks to the C^1 continuity of Poisson regions, our algorithm is adequate to produce smooth local details.

To render vector graphics, Orzan et al. [5] designed a multigrid solver to use a coarse version of the sections. Jeschke et al. [8] used finite differences with varying step size to accelerate the calculation process. Boye et al. [9] presented a finite element method (FEM) which can convert a diffusion curve im-

*Corresponding author
Email address: YHe@ntu.edu.sg (Ying He)

age into a mesh-based representation. It may be noted that, a *global* solver is not necessary for mesh rendering, since only faces within or adjacent to the painting/decoration region shall be further processed. Based on the vector solver of PVG, we devise a *local* Poisson solver to render mesh details efficiently, while avoiding to process global details. With the local solution, only the vertices that contribute to the results are evaluated. Our contributions are as follows:

- At the theoretic and computational front, we propose an efficient Poisson solver directly on 3D meshes, which is a local solver for rendering with more flexibility.
- In the application of 3D decorations, PVG provides users with simple inputs (i.e., a set of sparse curves) to produce detailed geometries. In particular, smooth 3D decorations could directly benefit from PR within the framework of PVG, which could not be properly enabled by using conventional DCs.
- Thanks to the smoothness property inherited from PR, our new method will afford level-of-details editing directly on complex geometries.

The remaining of the paper is organized as follows. Section 2 reviews the related works and Section 3 introduces the preliminary knowledge and required terminologies on Poisson vector graphics and harmonic B-splines. Section 4 presents our PVG mesh solver in details and Section 5 addresses our specific algorithm, followed by experimental results and discussions in Section 6. Finally, Section 7 concludes the paper and points out the future work.

2. Related Work

This section briefly reviews the work relevant to ours.

2.1. Surface Decoration

Besides texture mapping, there are a few other alternatives for decorating 3D surfaces. Bump mapping [10, 11, 12] simulates 3D surface details by perturbing normal vectors while the original geometry remains unchanged. In contrast to bump mapping that can only control shading normals, parallax mapping [13, 14] can also modify the texture coordinates along the view direction to obtain mesostructure normals and color data based on the depth values using an approximate solution. Relief texture mapping [15, 16] uses textures enhanced with depth information to create the illusion of complex geometric details onto flat polygons with locating the intersection of the height field and the ray to generate pixel shader. In contrast to bump mapping, displacement mapping [17, 18, 19] actually changes model's geometry by moving vertices along surface normals.

Using texture and displacement mappings, Donnelly [20] synthesized detailed geometry on the surfaces of objects by mapping the geometric texture. Zhou et al. [21] proposed a mesh quilting algorithm to seamlessly synthesize geometric texture sample inside a thin shell around an arbitrary surface through local stitching and deformation. Lai et al. [22] converted the 2D texture into geometric image to reconstruct synthesized geometry. Bhat et al. [23] presented an example-based method for synthesizing geometric textures such as pits and grooves on surfaces by extending the neighborhood-based texture synthesis algorithms for volumetric models. By reconstructing the fine-scale geometric details over a simple proxy of the original model, Toledo et al. [24] proposed a suitable representation for highly tessellated models using a set of geometry textures. Landreneau and Schaefer [25] developed a method for generating scales and scale-like structures on a polygonal mesh. Taking a user-sketched lateral line as input (which controls the distribution and orientation of scales), it computes a vector field over the surface to control an anisotropic centroidal Voronoi tessellation and then automatically fills each Voronoi cell with oriented scales and generates a fully connected 2-manifold mesh that is suitable for subsequent post-processing applications. Their method is effective to generate pre-defined patterns repetitively on 3D surfaces, while our method supports free-form vector graphics and can also provide the user more control of the displacement.

2.2. Vector Graphics

Diffusion curves, proposed by Orzan et al. [5], are 2-sided curves with colors defined on either side. By diffusing these colors over the image, the resulting image includes sharp boundaries along the curves with smoothly shaded regions between them. Note that diffusion curve images are harmonic functions of colors, the maximum principle states that a non-constant harmonic function cannot attain a maximum (or minimum) at an interior point of its domain. Therefore, the follow-up work focuses on providing more degrees of freedom for controlling color gradient. Bezerra et al. [26] proposed diffusion barriers, diffusion anisotropy, and spatially varying color strength to control the diffusion process. Their approach is able to diffuse both colors and normal maps, hereby producing interesting non-photorealistic effects. However, it is non-intuitive to specify the boundary condition for normals and normals can only be diffused within a closed diffusion curve. Using thin-plate splines (TPS), Finch et al. [27] extended diffusion curves to provide smooth interpolation through color constraints. Although TPS allows more user control and is able to mimic smooth shading, it often produces unwanted local extremals (hereby unpredicted colors) due to the violation of the maximal principle of harmonic equation. Moreover, solving a bi-Laplace's equation is more computationally expensive than solving Laplace's equation, and it may suffer from serious numerical issues since the system is less well-conditioned. Lieng et al. [28] proposed shading curves, which associate shading profiles to each side of the curve. These shading profiles, which can be manually manipulated, represent the color gradient out from their associated curves. Lecot and Levy [29] developed a vectorization method which is based on a two-level variational parametric segmentation algorithm, minimizing Mumford and Shah's energy and operating on an intermediate triangulation, well adapted to the features of the image. Using holomorphic 1-form, Lai et al.

[30] developed an automatic method for converting raster images to high-quality gradient meshes with non-trivial topologies. Xie et al. [31] automatically generated sparse diffusion curve vectorizations of raster images by fitting curves in the Laplacian domain. Their method is highly efficient, combines Laplacian and bi-Laplacian diffusion curve representations, and generates a hierarchical representation that accurately reconstructs both vector art and natural images.

3. Preliminary

In this section, we briefly introduce harmonic B-splines, diffusion curves and Poisson vector graphics. For details, we refer readers to [32, 7]. To ease reading, we list the main notations in Table 1.

Item	Description
Ω	2D domain
$\mathcal{M}$	Input 3D triangle mesh
$U \subset \mathcal{M}$	A simply connected region on the mesh
$\mathbf{x}$, $\mathbf{y}$	Vertices of Ω or $\mathcal{M}$
$\mathbf{n}$	Normal vector
$\mathbf{t}_i$	The i-th knot of Ω or $\mathcal{M}$
$\bigcup_i \mathcal{V}_i$	Voronoi diagram
$N_1(\mathcal{V}_i)$	1-ring neighboring cells of $\mathcal{V}_i$
n	Number of knots (Voronoi cells)
$A_{\mathcal{M}}$	The total area of mesh $\mathcal{M}$
$A_{\mathcal{V}_i}$	The area of Voronoi cell $\mathcal{V}_i$
$A_{\mathbf{x}}$	The area of Voronoi region of vertex $\mathbf{x}$
e_{ij}	Voronoi edge between cells $\mathcal{V}_i$ and $\mathcal{V}_j$
d_{ij}	Geodesic distance between $\mathbf{t}_i$ and $\mathbf{t}_j$
f	Laplace constraints of PCs and PRs
g	The Dirichlet boundary condition
$u(\mathbf{x})$	The solution of Poisson equation
$\triangle$	The Laplace-Beltrami operator
$\phi_{\mathbf{y}}(\mathbf{x})$	Green's function centered at vertex $\mathbf{y}$

Table 1: Notations

3.1. Harmonic B-splines

Let $\Omega \subset \mathbb{R}^2$ be a 2D compact domain and $\mathcal{T} = \{\mathbf{t}_i | \mathbf{t}_i \in \Omega\}_{i=1}^n$ a set of knots. Taking $\{\mathbf{t}_i\}$ as the generators, we construct a Voronoi diagram $\Omega = \bigcup_{i=1}^n \mathcal{V}_i$, where $\mathcal{V}_i$ is the Voronoi cell of knot $\mathbf{t}_i$.

For arbitrary points $\mathbf{x}, \mathbf{y} \in \Omega$, Green's function of the Laplace operator Δ satisfies

$$\Delta \phi_{\mathbf{y}}(\mathbf{x}) = \delta_{\mathbf{y}}(\mathbf{x}), \tag{1}$$

where $\delta_{\mathbf{y}}(\mathbf{x})$ is the Dirac delta function centered at $\mathbf{y}$. In 2D space, an analytic solution to Equation (1) is

$$\phi_{\mathbf{y}}(\mathbf{x}) = \frac{1}{2\pi} \log(\|\mathbf{x} - \mathbf{y}\|).$$

For a Voronoi cell $\mathcal{V}_j$, applying Green's theorem to (1) yields

$$\int_{\mathcal{V}_j} \Delta \phi_{\mathbf{y}}(\mathbf{x}) \, \mathrm{d}\sigma = \int_{\partial \mathcal{V}_j} \frac{\partial \Delta \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} \, \mathrm{d}s, \tag{2}$$

where $\mathbf{n}$ is the outward unit normal to the boundary $\partial \mathcal{V}_j$, $\mathrm{d}\sigma$ and $\mathrm{d}s$ are the area and line integral elements, respectively.

Then we define a function ψ_j for each Voronoi cell $\mathcal{V}_j$ as

$$\psi_j(\mathbf{x}) = \sum_i w_{ij} \phi_{\mathbf{t}_i}(\mathbf{x}), \tag{3}$$

where w_{ij} is the discrete Laplacian weight and $\sum_i w_{ij} \phi_{\mathbf{t}_i}(\mathbf{x})$ is a boundary sum that approximates the line integral on the right hand side of Equation (2).

Feng and Warren [32] showed that the functions ψ_j are *approximately* local, nonnegative, and satisfying partition of unity, sharing many properties of the popular B-spline's basis functions. Therefore they named it *harmonic* B-spline. Harmonic B-spline has two salient features that distinguish itself from the conventional B-spline. First, there is no restriction on knots and splines can be constructed on a set of fully irregular knots. Second, it is parameterization free, since evaluating Green's function $\phi_{\mathbf{y}}(\mathbf{x})$ requires only the distance $\|\mathbf{x} - \mathbf{y}\|$, which can be measured in a coordinate-free manner.

3.2. Diffusion Curves & Poisson Vector Graphics

Orzan et al. [5] developed diffusion curves, which are 2-sided curves with colors defined on either side. By diffusing these colors over the image, the final result includes sharp boundaries along the curves with smoothly shaded regions between them (see Figure 1(a)(b)). Mathematically speaking, diffusion curve image is the solution of the following Laplace equation $\triangle u(\mathbf{x}) = 0$, $u(\mathbf{x})|_{\partial\Omega} = g$, where g is user-specified boundary color. Since the solution of Laplace equation is a harmonic function, which, by the maximum principle, retains its maximum and minimum on the domain boundary, implying that u cannot have local maxima or minima, unless it is constant. As a result, diffusion curves do not have enough degrees of freedom to control color gradient.

Poisson vector graphics [33] extends the DC framework by adding two new geometric primitives, called Poisson curves (PC) and Poisson regions (PR). The former is to model color discontinuity across curves, while the latter is to design smooth shading within the user specified regions. PVG solves a Poisson's equation with piecewise constant Laplacians f, hereby taking DC as a special case with $f \equiv 0$. Extending the zero Laplacian to a piecewise constant function f brings 4 unique advantages. First, users can explicitly control the local and/or global shading profiling via manipulating f (which is a scalar for each color channel). Second, users can easily control the color extrema, which are either on the curves (for PC and DC) or inside a region (for PR). Third, PVG allows intersection among the geometric primitives. Fourth, PVG natively supports seamless cloning. Although a PVG can have an arbitrary number of PCs and PRs, it must contain at least one diffusion curve, serving as the boundary condition g. Hou et al. [33] demonstrated that PVG is a simple yet powerful authoring tool that can produce photo-realistic vector graphics from scratches. See Figure 1(c)(d) for a simple example of PVG.

(a) DCs (b) DC image (c) DCs & PRs (d) PVG image

Figure 1: Diffusion curves and Poisson vector graphics [33]. PVG complements DC by two new primitives, Poisson regions and Poisson curves, which provide additional control on colors.

3.3. 2D PVG Solver

To render PVG, one solves the following Poisson equation

$$\Delta u(\mathbf{x}) = f, \quad u(\mathbf{x})|_{\mathbf{x}\in\partial\Omega} = g, \tag{4}$$

where f is the Laplacian constraint of Poisson curves and regions, and g is the Dirichlet boundary condition of colors. Using harmonic B-splines [32], Hou et al. [33] developed a novel random-access solver for the above Poisson equation.

Green's third identity provides an analytical solution for Laplace/Poisson's equations in the form of integral,

$$\begin{aligned} u(\mathbf{x}) &= \iint_{\Omega} \phi_{\mathbf{y}}(\mathbf{x})\Delta u(\mathbf{y}) d\sigma_{\mathbf{y}} \\ &+ \oint_{\partial\Omega} \left(u(\mathbf{y}) \frac{\partial \phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} - \phi_{\mathbf{y}}(\mathbf{x}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}} \right) dl_{\mathbf{y}}, \end{aligned} \tag{5}$$

where $d\sigma$ and dl are the surface and line elements, $\mathbf{n}$ is the outward pointing unit normal of dl.

Hou et al. [33] rasterized the image domain Ω with user-specified resolution and then discretized the geometric primitives (DCs/PCs/PRs) using quad-tree. Taking the quad-tree nodes as generators, they computed a Voronoi diagram to partition the domain $\Omega = \bigcup_i \mathcal{V}_i$. For each Voronoi cell $\mathcal{V}_i$, they constructed a harmonic B-spline basis function $\psi_i(\mathbf{x})$, whose knots are the generators of $\mathcal{V}_i$ and its 1-ring neighboring cells.

For basis function of a boundary Voronoi cell $\mathcal{V}_i$, they set its control point λ_i using the Dirichlet boundary condition g. For basis functions of internal Voronoi cells, they computed their control points by solving a sparse linear system of size $k \times k$, where k is the number of interior quad-tree nodes. Finally, the solution is given by $u(\mathbf{x}) = \sum_{i=1} \lambda_i \psi_i(\mathbf{x})$. Since harmonic B-spline basis functions are approximately local, evaluating $u(\mathbf{x})$ involves only the basis functions close to $\mathbf{x}$.

4. PVG Mesh Solver

Let $\mathcal{M}$ be a closed manifold mesh and $U \subset \mathcal{M}$ a simply connected region enclosing the PVG primitives. To render PVG on meshes, we adopt the harmonic B-spline based solver similar to [33]. For a closed mesh $\mathcal{M}$, Green's function of the Laplace-Beltrami operator is

$$\Delta \phi_{\mathbf{y}}(\mathbf{x}) = \delta_{\mathbf{y}}(\mathbf{x}) - \frac{1}{A_{\mathcal{M}}}, \tag{6}$$

where $A_{\mathcal{M}}$ is the area of $\mathcal{M}$. The addition of the area term is necessary to ensure the existence of a solution $\mathbf{y}(\mathbf{x})$ in the compact case. Using Green's third identity on closed manifolds, we can write the solution $u(\mathbf{x})$ as

$$u(\mathbf{x}) = \frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma + \iint_U \Phi_{\mathbf{y}}(\mathbf{x}) \Delta u(\mathbf{x}) d\sigma \tag{7}$$

Comparing to the planar case (Equation (5)), we have the additional term $\frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma$ which is the average of $u(\mathbf{x})$ in region U. Such a term reveals the difference of Green's function defined on closed and open domains.

We select a set of knots $\{\mathbf{t}_j\}_{j=1}^n$ which is a subset of mesh vertices and construct a Voronoi diagram on $\mathcal{M}$. The knot selection criteria are as follows: (1) the outer boundary of $\mathcal{V}$ coincides with the boundary of U. (2) the regions on the boundary (i.e., $\partial\mathcal{V}_j \cap \partial U \neq \Phi$) are small enough. (3) the function f in every $\mathcal{V}_j$ is a constant. Similar to [7], we discretize $u(\mathbf{x})$ as

$$u(\mathbf{x}) \approx \frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma + \sum_{\mathcal{V}_j} A_{\mathcal{V}_j} f|_{\mathcal{V}_j} \bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) \tag{8}$$

Feng and Warren [32] defined the basis function for Voronoi cell $\mathcal{V}_j$ as

$$\psi_j(\mathbf{x}) = \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij} (\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x}))$$

where the coefficient $a_{ij} = -\frac{\|e_{ij}\|}{d_{ij}}$, and the Voronoi edges e_{ij} are fallen into three disjoint classes geometrically: inner boundaries $\mathcal{E}_{ib}$, outer boundaries $\mathcal{E}_{ob}$ and interior edges $\mathcal{E}_{ie}$.

Then we can derive

$$\begin{aligned} &\sum_{\mathcal{V}_j} \lambda_j \psi_j \\ &= \sum_{\mathcal{V}_j} \lambda_j \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{e_{ij}\in\mathcal{E}_{ob}} a_{ij} (\bar{\phi}_{\mathcal{V}_i} - \bar{\phi}_{\mathcal{V}_j}) \lambda_j \\ &= \sum_{\mathcal{V}_j} \lambda_j \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_j \in \mathcal{I}} \left(\bar{\phi}_{\mathcal{V}_j} \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij} (\lambda_i - \lambda_j) \right) \end{aligned} \tag{9}$$

For λ_j of $\mathcal{V}_j \in \mathcal{B}$, we simply set $\lambda_j = g|_{\mathcal{V}_j}$. To compute the coefficients of $\bar{\phi}_{\mathcal{V}_j}$ ($\mathcal{V}_j \in \mathcal{I}$) satisfying $A_{\mathcal{V}_j} f|_{\mathcal{V}_j}$, we solve the following equation

$$\mathbf{L}^{\mathcal{I}} \boldsymbol{\lambda}^{\mathcal{I}} = \mathbf{b} - \mathbf{L}^{\mathcal{B}} \boldsymbol{\lambda}^{\mathcal{B}}, \tag{10}$$

where $b_j = A_{\mathcal{V}_j} f|_{\mathcal{V}_j}$. It is apparent that λ_j is the average color or displacement in $\mathcal{V}_j$. It is easily verified that the third and fourth terms of Eqn. (9) are identical to inner boundaries and outer boundaries of Eqn. (8). Since λ_j is the average color or displacement of $\mathcal{V}_j$, $\sum_{\mathcal{V}_j \in \mathcal{I} \cup \mathcal{B}} \lambda_j \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}}$ is the average color/displacement of region U. Thus, the Eqn. (9) is the solution of the Poisson equation (4).

Since the basis function $\psi_j(\mathbf{x})$ decays to zero quickly, we evaluate the $u(\mathbf{x}) = \sum_{\mathcal{V}_j \in \mathcal{N}(\mathbf{x})} \lambda_j \psi_j(\mathbf{x})$ using only the basis functions in a local neighborhood of $\mathbf{x}$.

Remark. Our solver can be extended to handle open meshes. By removing the constant term and adding the boundary integral to Eqn. (7), we can express the solution as

$$u(\mathbf{x}) = \frac{1}{A_{\mathcal{M}}} \iint_U u(\mathbf{x}) d\sigma + \iint_U \Phi_{\mathbf{y}}(\mathbf{x}) \Delta u(\mathbf{x}) d\sigma + \oint_{\partial U} \left(u(\mathbf{x}) \frac{\partial \Phi_{\mathbf{y}}(\mathbf{x})}{\partial \mathbf{n}} - \Phi_{\mathbf{y}}(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial \mathbf{n}} \right) dl. \tag{11}$$

The basis function for Voronoi cell $\mathcal{V}_j$ is

$$\psi_j(\mathbf{x}) = \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij} (\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})),$$

so we obtain the solution $u(\mathbf{x}) = \sum_{\mathcal{V}_j \in \mathcal{N}(\mathbf{x})} \lambda_j \psi_j(\mathbf{x})$ in a similar form as the case of closed meshes.

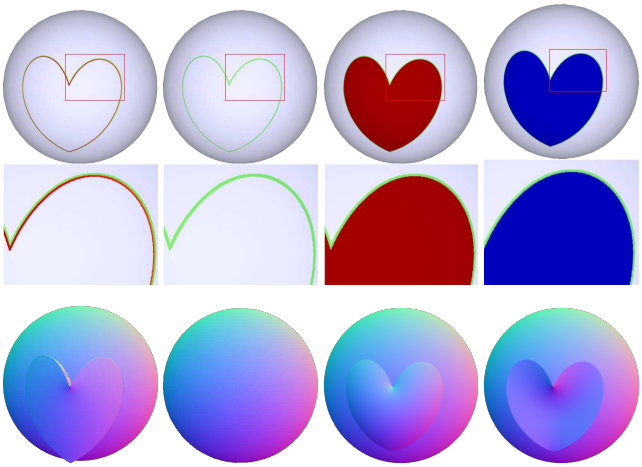
5. Algorithm

5.1. Overview

PVG consists of three types of primitives: diffusion curve, Poisson curve and Poisson region. DC is a curve with color (cDC) or displacement (dDC) specified by the user, meanwhile, it guarantees a unique solution of the Poisson solver as Dirichlet boundary condition. DC is like a radiative heat transport diffusing color or height on the mesh. PC is a double-sided curve with opposite Laplacian values and can create high contrast effect along the curve. PR is a region whose boundary and inside region have reverse Laplacians and can generate a controllable soft boundary there. PR is effective to produce the effects like highlight and smooth convex or concave bump effect. In short, DCs specify color and height on the mesh as the boundary condition, PCs and PRs control tone by offsetting Laplacians.

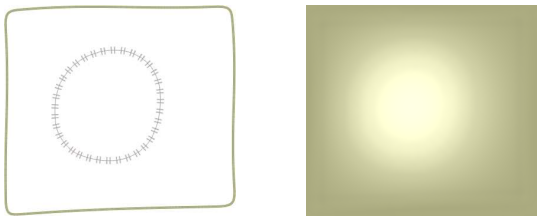
In this paper, we adopt diffusion curves and Poisson regions to represent both colors and displacements on 3D surfaces, and use prefixes "c" and "d" to distinguish them. For example, cDC is a color field represented by diffusion curves and dPR is displacement generated by Poisson regions. Figure 5 shows a simple example with 5 cDCs, 4 dDCs and 1 dPR. Besides, from Fig. 5(e)(g), we can see that how PRs make the difference from traditional DCs to achieve smooth tune result. As shown in Fig. 2, DC alone produces only sharp feature, whereas DC+PR can make a smooth displacement field. Furthermore, PRs are C^1-continuous across their boundaries for 2D cases, thanks to the analytic solution of Green's functions (see Fig. 3). Unfortunately, we don't have such a luxury for general 3D meshes, the C^1-continuity does not hold any longer. Nevertheless, editing PRs is still more flexible than DCs, since PRs can intersect with each other, whereas DCs cannot (see Fig. 4).

Figure 6 illustrates the pipeline of our surface decoration algorithm. Taking the user-sketched curves as input, we take a subset of mesh vertices as knots and then construct a geodesic Voronoi diagram. Then, for each Voronoi cell, we construct a harmonic B-spline basis function ψ. By solving small sparse linear system, we obtain the control coefficients λ and finally express the color/displacement function using $\sum_i \lambda_i \psi_i$.



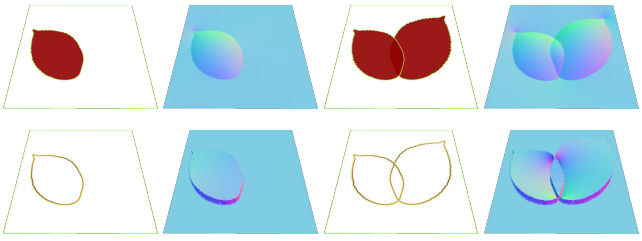
(a) dDC $f \neq 0$ dPR $g = 0$ (b) dDC $f = 0$ dPR $g = 0$ (c) dDC $f = 0$ dPR $g = 1$ (c) dDC $f = 0$ dPR $g = -1$

Figure 2: Diffusion curves vs Poisson regions. Diffusion curves are associated with the Dirichlet boundary condition f, while Poisson regions are assigned Laplace constraints g. We solve the Poisson equation $\triangle u = g$ and $u|_{\partial\Omega} = f$ to compute the displacement field. (a) Although the displacement generated by DC is smooth within Ω, it is discontinuous along the domain boundary $\partial\Omega$. (b) The only exception is $f \equiv 0$, which produces a constant function $u \equiv 0$. (c)-(d) Combining DC and PR, we can define displacement which is C^1 continuous both inside Ω and across the boundary $\partial\Omega$. We visualize the values of f and g using color maps: warm colors are positive values and cold colors are negative values, and green is 0.



(a) PVG Primitives (b) PVG image

Figure 3: On 2D domain, PR (dashed curve) is C^1 continuous across its boundary, thanks to the analytical solution of Green's functions of the 2D Laplacian operator.



(a) (b) (c) (d)

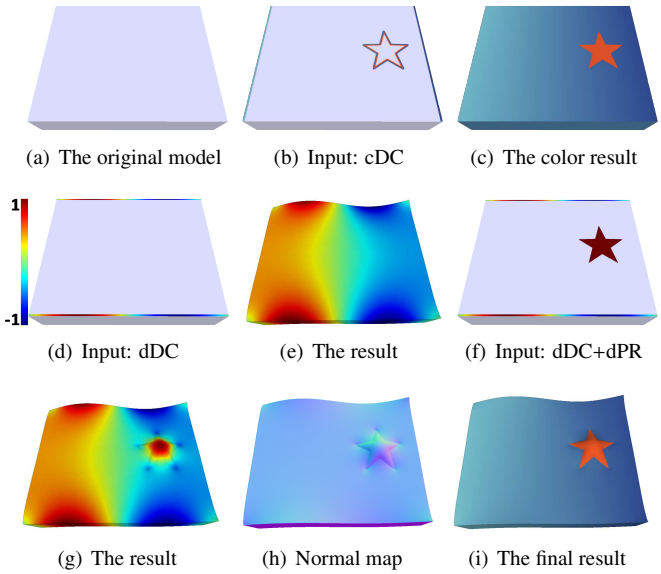
Figure 4: PVG allows intersecting primitives, which is a desired feature for editing. Row 1 shows two PRs intersecting each other. When two diffusion curves are intersecting, their associated boundary conditions are often "competing", resulting in non-smooth artifacts (see row 2).

Algorithm 1 Decorating 3D meshes with PVG

Require: The triangle mesh $\mathcal{M}$, the PVG primitives (i.e., diffusion curves and Poisson regions) associated with color and/or displacement constraints

Ensure: The mesh with details and colors decorated by PVG

Compute discrete Green's function $\phi_{\mathbf{y}}(\mathbf{x})$ using [34]
Sample knots $\{\mathbf{t}_i\}_{i=1}^n$ from mesh's vertices
Construct geodesic Voronoi diagram on $\mathcal{M}$, and $\mathcal{M}$ is divided into a set of sub-regions $\mathcal{M} = \bigcup_{i=1}^n \mathcal{V}_i$
// Construct the harmonic B-spline basis function $\{\psi_i\}_{i=1}^n$
for each Voronoi cell $\mathcal{V}_j$ **do**
    $\psi_j(\mathbf{x}) = \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x}))$
    $\bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) = \frac{1}{A_{\mathcal{V}_j}} \sum_{\mathbf{y} \in \mathcal{V}_j} A_{\mathbf{y}} \phi_{\mathbf{y}}(\mathbf{x})$
end for
// Compute the control points $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^n$
for each Voronoi cell $\mathcal{V}_i$ **do**
    Compose i-th row of sparse matrix $\mathbf{L} = \{a_{ij}\}_{i,j=1}^n$
    for every neighbor $\mathcal{V}_j \in N_1(\mathcal{V}_i)$ **do**
        $a_{ij} = -\frac{\|e_{ij}\|}{d_{ij}}$
    end for
    $a_{ii} = \sum_{\mathcal{V}_j \in N_1(\mathcal{V}_i)} a_{ij}$
    $b_i = A_{\mathcal{V}_i} f|_{\mathcal{V}_i}$
end for
Solve $\mathbf{L}\boldsymbol{\lambda} = \mathbf{b}$
// Compute the decorated mesh
for every vertex $\mathbf{x} \in \mathcal{V}_i$ **do**
    $u(\mathbf{x}) = \sum_{\mathcal{V}_j \in N(\mathbf{x})} \lambda_j \psi_j(\mathbf{x})$
end for



(a) The original model (b) Input: cDC (c) The color result
(d) Input: dDC (e) The result (f) Input: dDC+dPR
(g) The result (h) Normal map (i) The final result

Figure 5: Generating a flag model from a cuboid model with simple input based PVG

5.2. Computing Discrete Green's Function

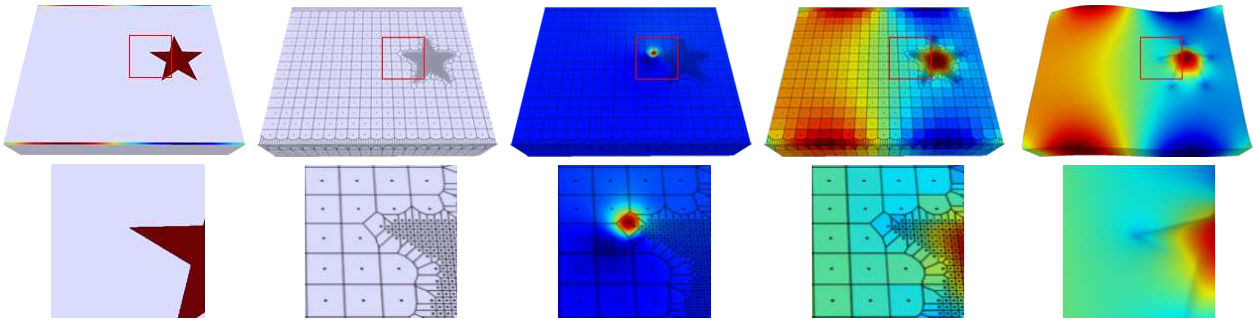
Define $g_{ij} = \phi_{\mathbf{v}_i}(\mathbf{v}_j)$ the Green's function centered at vertex $\mathbf{v}_i$ evaluating at vertex $\mathbf{v}_j$. We denote by $\mathbf{G} = (g_{ij})_{m\times m}$ the discrete Green's function on closed mesh M, where m is the number of vertices. Lipman et al. [34] proved that $\mathbf{A}^{-1}\mathbf{L}\mathbf{G} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, where $\mathbf{1}$ is the column vector of all ones. Denote by $\mathbf{M}_j$ the j-th column of matrix $\mathbf{M}$. They showed that $\mathbf{G}_j = \mathbf{x} - \frac{\mathbf{1}^T\mathbf{x}}{\mathbf{1}^T\mathbf{1}}\mathbf{1}$, where $\mathbf{x}$ is a particular solution to $\mathbf{L}\mathbf{A}^{-1}\mathbf{L}\mathbf{x} = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)_j$. To get the particular solution, replace the first row and the first column of $\mathbf{A}^{-1}\mathbf{L}$ by zeros and set the diagonal entry at their intersection to 1. Also replace the first row of $(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)_j$ by 0. As pointed out in [34], this kind of linear system can be solved very efficiently by first performing Cholesky factorization of $\mathbf{L}\mathbf{A}^{-1}\mathbf{L}$ and then performing two backward substitutions for every given vector on the right hand side. Green's functions on open mesh can be similarly solved by $\mathbf{A}^{-1}\mathbf{L}\mathbf{G} = \mathbf{I}$.

5.3. Computing Geodesic Voronoi Diagram

While user drawing DC and PR's boundary on the mesh, each curve assigned with double-side color or Laplacian weight are represented by a B-spline curve. According to the work in [35], De Casteljau's algorithm are generalized into 2D manifold which replaced the linear interpolation with the smooth geodesic interpolation. With this framework, we represent PVG primitives by B-spline curves on the mesh. In actual, there need massive computation of pairwise geodesic path. Here we adopt an efficient approach [36] to query approximate geodesic path between any two points on the mesh. After given the sketch based B-spline curves, we find the vertices belonging to the B-spline curves and assign them with corresponding color or Laplacian.

To guarantee the accuracy of computation, we need sample the knots appropriately to satisfy the condition that the Laplacian in every Voronoi cell region is constant. At first, all key vertices involved in the input PVG are included in the knot set. And then some other vertices are evenly selected additionally with the specified knot density. Next, these knots are served as generators to construct a geometric Voronoi diagram on the mesh.

The algorithm of building Voronoi diagram in $\mathbb{R}^2$ has been well implemented such as by using dual Delaunay triangulation. But in $\mathbb{R}^3$, the situation becomes more complicated and one of the most differences is that the bisector is a point trace on the mesh which have equal geodesic distance to two knot neighbors rather than a straight line. First, the knots are used as sources to create multi-source exact geodesic distance field on the mesh by applying the improved CH algorithm [37]. After that, each vertex is assigned a geodesic distance to its closest knot. Then, we check each triangle edge of the mesh and label it if its endpoints have different closest knots. And if a labeled edge's incident triangle face involves three labeled edges, it contains a branch point inside, otherwise it is crossed through by a segment of bisector. Finally, based on the labeled edges and faces, we obtain the geodesic Voronoi diagram (Fig. 6(b)) on the mesh through the marching algorithm [38].



(a) Displacement constraints (b) Knots $\{\mathbf{t}_i\}$ & Voronoi diagram $\{\mathcal{V}_i\}$ (c) Basis function $\psi_j(\mathbf{x})$ (d) Control points $\{\lambda_i\}$ (e) Harmonic B-spline function $u(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \psi_i(\mathbf{x})$

Figure 6: Illustration of our Poisson solver on the box mesh $\mathcal{M}$. (a) shows the domain U (because dDCs are not closed, here $U = \mathcal{M}$) and the displacement constraints including one dPR (red star) and four dDCs (another two on the bottom) along edges of the cuboid model, dPR assigns Laplacian constraints f and dDCs give the Dirichlet boundary condition g. (b) First, to guarentee that the Laplacian in every Voronoi cell is constant, sample n knots consisted of all the key vertices involved in the dPR and dDCs and some other vertices evenly distributed with specified knot density. Then use these interior and boundary knots as generators to construct geodesic Voronoi diagram, and U is divided into a set of disjoint sub-regions, $U = \bigcup_{i=1}^{n} \mathcal{V}_i$. (c) For a Voronoi cell $\mathcal{V}_j$, based on the Green's function of the Laplacian operator on $\mathcal{M}$, define a harmonic B-spline basis function $\psi_j(x)$, whose knot $\mathbf{t}_j$ is the generator of $\mathcal{V}_j$. (d) If $\mathcal{V}_j$ is a boundary Voronoi cell, simply set its control point λ_j using the given boundary condition g. If $\mathcal{V}_j$ is an internal Voronoi cell, compute its control point λ_j by solving a sparse linear system, whose size is much smaller than the number of vertices of the mesh. (e) After obtaining basis function and control points, construct harmonic B-spline function $u(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \psi_i(\mathbf{x})$ to give the solution of Poisson equation. As (c) shown that the basis function decays to zero quickly, we can evaluate the $u(\mathbf{x}) = \sum_{\mathcal{V}_i \in \mathcal{N}(\mathbf{x})} \lambda_i \psi_i(\mathbf{x})$ just using the basis in a neighbor around $\mathbf{x}$.

5.4. Computing Control Coefficients

According to the PVG constraints defined on the vertices and the constructed geodesic Voronoi diagram, we attempt to solve a sparse linear system to require control coefficients λ for expressing the harmonic B-spline function in the further step. As for constructing this linear system

$$\mathbf{L}\boldsymbol{\lambda} = \mathbf{b} \tag{12}$$

here $\mathbf{L} = \{a_{ij}\}$ is a sparse symmetric matrix whose size is $n \times n$ (n is the number of knots $\{\mathbf{t}_i\}$), and b is a n-dimensional vector defined as each knot's Laplacian value multiplying by its area. Based on the position relationships of knots in Voronoi diagram, they can be obtained specifically as follows:

$$a_{ij} = \begin{cases} -\frac{\|e_{ij}\|}{d_{ij}}, & if\ i \neq j\ and\ \{t_i, t_j\}\ are\ neighbors \\ 0, & if\ i \neq j\ and\ \{t_i, t_j\}\ are\ not\ neighbors \\ \sum_{\mathcal{V}_j \in N_1(\mathcal{V}_i)} a_{ij}, & if\ i = j;\ i, j \in [1, n] \end{cases} \tag{13}$$

$$b_i = A_{\mathcal{V}_i} f|_{\mathcal{V}_i}, i \in [1, n] \tag{14}$$

where a_{ij} $(i, j \in [1, n])$ is only related to the i-th knot $\mathbf{t}_i$ and its 1-ring Voronoi neighbourhood. Note that no matter how the knots are distributed, on *average*, each knot has six neighbors (since the average neighboring Voronoi cells is 6). Therefore, for each row of $\mathbf{L}$, there are roughly seven non-zero values.

Moreover, allow for $rank(\mathbf{L}) = (n - 1)$, and this linear system Eqn. (12) has infinitely many solutions, we use DCs to assign the Dirichlet boundary condition. By moving them to the righthand side of this equation as Eqn. (10), we can attain a unique solution. That is, the average color (height) of each Voronoi cell is obtained (Fig. 6(d)), which will serve as the control coefficients for the next step of constructing harmonic B-spline function. In addition, we know that any input of PVG needs to include at least one DC to guarantee the unique solution.

5.5. Constructing Basis Functions

In this paper, we would like to supply a good decoration method of the 3D meshes affecting the vertices' features in both color and geometry aspects, making that the Poisson solver is working on all vertices of the model. Of course, the more dense mesh model you use and the higher resolution result you get. To achieve a good rendering result satisfying the resolution requirement, the original mesh model needs enough vertex density. By the way, in our interface, we use midpoint subdivision to increase the model's density if it is not dense enough.

Therefore, the fundamental problem of the Poisson solver is to reduce the computation burden which matters a lot especially for the large meshes we need. Here, we propose an efficient method of solving the Poisson solver on the dense mesh through constructing harmonic B-spline function. Harmonic B-spline function provides the local computation for solving the Poisson equation on the mesh instead of a global solution covering all vertices as the conventional finite element method does.

As for the construction of harmonic B-spline function on the 3D mesh $\mathcal{M}$, the above computation has supplied its control coefficients. And according to Green's third identity theory and previous derivation, the j-th basis function of harmonic B-spline can be defined on the closed mesh by

$$\psi_j(\mathbf{x}) = \frac{A_{\mathcal{V}_j}}{A_{\mathcal{M}}} + \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})), j \in [1, n] \tag{15}$$

and which is on the open boundary mesh:

$$\psi_j(\mathbf{x}) = \sum_{\mathcal{V}_i \in N_1(\mathcal{V}_j)} a_{ij}(\bar{\phi}_{\mathcal{V}_i}(\mathbf{x}) - \bar{\phi}_{\mathcal{V}_j}(\mathbf{x})), j \in [1, n] \tag{16}$$

where $\mathcal{V}_i$ is 1-ring neighbour of $\mathcal{V}_j$, $A_{\mathcal{M}}$ is the total area of the mesh, $A_{\mathcal{V}_j}$ is the area of j-th Voronoi cell $\mathcal{V}_j$, and is the average value of Green function ϕ in Voronoi cell $\mathcal{V}_j$:

$$\bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) = \frac{1}{A_{\mathcal{V}_j}} \int_{\mathcal{V}_j} \phi(\mathbf{x}, \mathbf{y})\, d\delta_{\mathbf{y}} \tag{17}$$

whose discrete form can be expressed as

$$\bar{\phi}_{\mathcal{V}_j}(\mathbf{x}) = \frac{1}{A_{\mathcal{V}_j}} \sum_{\mathbf{y} \in \mathcal{V}_j} A_{\mathbf{y}} \phi_{\mathbf{y}}(\mathbf{x}) \tag{18}$$

Here $\phi_{\mathbf{y}}(\mathbf{x})$ is the Green function on the mesh, which can be obtained by the method presented in [34]. Green function on a specific 3D mesh only needs to be calculated once, which is attained in the preprocessing step in our interface.

And then we can derive the harmonic B-spline function as follows:

$$u(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \psi_j(\mathbf{x}) \tag{19}$$

where $u(\mathbf{x})$ is each vertex's color (height), except these $\mathbf{x}$ belonging to the knot set which $u(\mathbf{x})$ are already obtained (by Eqn. (12)) or known (specified by DCs) and no need to be calculated again, the other $u(\mathbf{x})$ are unknown and need to be calculated. And λ_j is the control coefficients required from Eqn. (12).

Since the basis function $\psi(\mathbf{x}, \mathbf{y})$ decays to zero quickly, we can evaluate Eqn. (19) by just using the basis of a neighbourhood around $\mathbf{x}$:

$$u(\mathbf{x}) = \sum_{\mathcal{V}_j \in N(\mathbf{x})} \lambda_j \psi_j(\mathbf{x}) \tag{20}$$

After solving this harmonic B-spline function Eqn. (20), each vertex's color/displacement can be obtained eventually (Fig. 6(e)). Besides, thanks to the random-access solver, we can flexibly and selectively decide the region of interest, hereby reducing the computational cost. For instance, applying the visibility algorithm such as back-face culling [39] or Z-buffer [40], we can check whether a point $\mathbf{x}$ is visible. If not, we don't need to compute $u(\mathbf{x})$ at all.

6. Results

We implemented our algorithm in C++ and evaluated it on a laptop with an Intel i7 CPU2.80GHz. Our method allows user to decorate 3D meshes in 3 ways: 1) user sketches PVG primitives (DCs, PCs and PRs) directly on the mesh, and then tunes the Dirichlet boundary condition g and Laplacian of colors/displacements g; 2) user attaches a pre-defined 2D PVG to the region of interest on the 3D model using *local* parameterization; and 3) user tessellates a pre-defined 2D PVG to the entire 3D model using global parameterization.

In Fig. 7, we decorate the Bunny model with the SPM logo and some colorful patterns. Using PVG, mesh decoration becomes easier, since users can sketch arbitrary curves on the 3D model directly and then tune the colors and displacements via setting the associated functions f and g. Besides, as the input affecting a small part of the mesh, our Poisson solver can provide a local computation method for the useful regions which lightens the calculation burden a lot in actual situations. As shown Fig. 7(d), based on the input, we can obtain a connected region with open boundary which need to be calculated. After building a geodesic Voronoi diagram with the n selected knots from that area, we can calculate n control points of harmonic B-splines function via equation $\mathbf{L}\boldsymbol{\lambda} = \mathbf{b}$. However, only vertices whose Voronoi cells contain input area (the green regions) will be actually calculated their $u(\mathbf{x})$.

Fig. 8 shows the result of a stele model decorated with concave Chinese calligraphy characters and convex patterns of auspicious clouds and dragon. In this stele decoration, first, we use two dDCs to generate a whole concave effect of the middle plaque with the inconsecutive displacement boundaries. Second, raising the whole dragon shape a bit by also using two dDCs. Third, smoothly embossing the middle part of the dragon and the auspicious clouds by using dPRs. Finally, smoothly carving the Chinese calligraphy characters on the stele mesh by using dPRs. With flexibly using different types of PVG, we can achieve various gorgeous effects in both coloring and embossment aspects.

Fig. 9 shows the result of a vase model decorated with blue and white porcelain patterns and smooth petal bumps. On one hand, the vase is decorated with blue and white porcelain patterns in the RGB field. On the other hand, all the petals are smoothly embossed on the mesh. From this example, we can see that our method is still powerful and available even if the input is delicate and exquisite.

Considering that the mesh need to be decorated by massive repetitive patterns sometimes, our system also accepts global parametrization as the PVG input to generate rendering result (see Fig.10). Furthermore, the combination of sketching and global parametrization can design various gorgeous decoration effects (see Fig. 11).

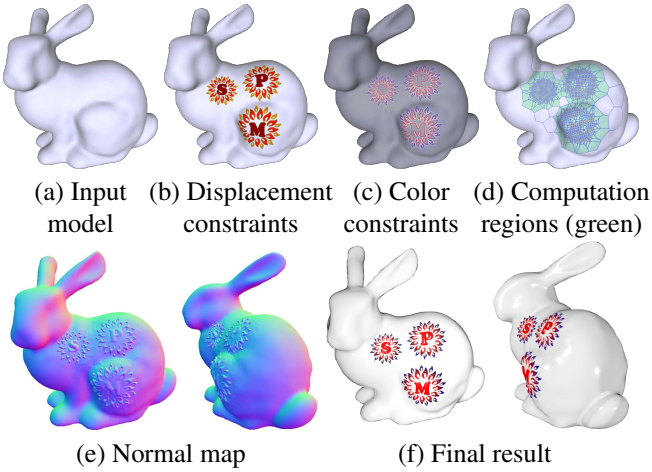
In our current implementation, the bottleneck is the precomputation stage, including constructing the geodesic Voronoi diagram and computing the Green's functions. For the Piggy Bank model with roughly 40 PVG primitives, pre-computing takes 138.4s (see Fig. 12). Local updating/editing PVG is much more efficient than re-computing for the entire model, since Voronoi diagram can be updated locally and we only need to re-compute the basis functions for the affected Voronoi cells. As shown in Fig. 12, changing colors takes only 0.758s, and editing the geometries of PRs takes 2.889s.

Thanks to the smoothness provided by PRs, our system also allows multi-level sculpting for the meshes (see Fig. 13).

In addition, the statistics table of these results is shown as Table. 2.

According to the results in these figures, PVG can decorate 3D meshes with various effects in the sides of color and geometry by sketching and global parametrization. Particularly, in the field of 3D geometric detail synthsization, compared with the result of the sharp displacement boundaries by only using DCs, our PRs can give a smooth bump effect.

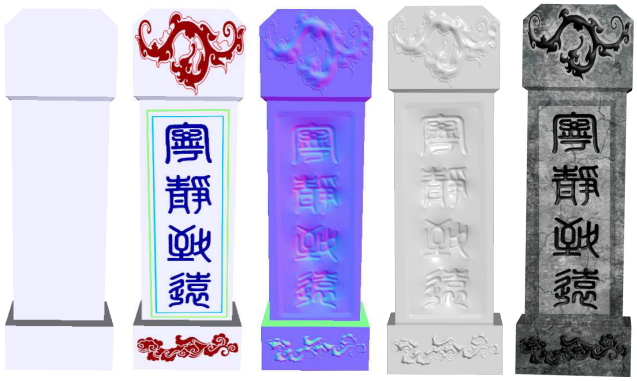
Jeschke [6] proposed a surface details decoration method



(a) Input model (b) Displacement constraints (c) Color constraints (d) Computation regions (green)

(e) Normal map (f) Final result

Figure 7: Bunny with the SPM logo. With a few diffusion curves and Poisson regions, we obtain colorful and smooth reliefs. Thanks to the *local* feature of our solver, we only need to construct the geodesic Voronoi diagram and harmonic B-splines for the region of interest.



(a) Input model (b) Displacement constraints (c) Normal map (d) Result w/o texture (e) Result w/ texture

Figure 8: Decorating the stele model with concave Chinese calligraphy characters and convex patterns of auspicious clouds and dragon. By simply adjusting the Laplacian values, we can control the height of the concave or convex reliefs. Compared the concave Chinese calligraphy characters and the convex patterns of auspicious clouds and dragon produced by dPR with those created by dDC, it is apparent that our PR could produce smooth details while DC could only produce sharp features. We visualize the displacement constraints by color map in (b), where warm and cold colors indicate positive and negative displacements respectively.

Triangle mesh	#cDC	#cPR	#dDC	#dPR
Bunny (Fig. 7)	82	0	82	81
Stele (Fig. 8)	0	0	22	31
Vase 1 (Fig. 9)	385	0	155	155
Vase 2 (Fig. 9)	264	0	264	264
Rhino (Fig. 10)	0	0	3038	3038
Sofa (Fig. 10)	27144	0	27144	27144
Cactus (Fig. 11)	854	0	122+854	122+854
Tree (Fig. 13)	$1+119\times2$	$1+119\times2$	$1+119\times2$	$1+119\times2$
Piggy bank (Fig. 12)	17(+3)	66(+15)	17(+3)	17(+3)

Table 2: Statistics. We report the number of diffusion curves and Poisson regions used. The prefixes "c" and "d" distinguish the primitives for color and displacement.

with diffusion curves. Since the C^0 continuity of Laplacian/Poisson's equations on boundary, the details are inevitable with sharp boundaries. Though the feature is pleasing sometimes, such as inlay ornaments on vase, smooth feature is pleasing for some other occasions, such as Figure 13. Our C^1 continuous PR provides a natural multilevel tool for editing complex decorations.

Finite element method [9] just evaluates C^0 continuity solution of Poisson's equations. Though high-order elements could result in smooth solution, the computational cost increases dramatically.

7. Conclusion & Future Work

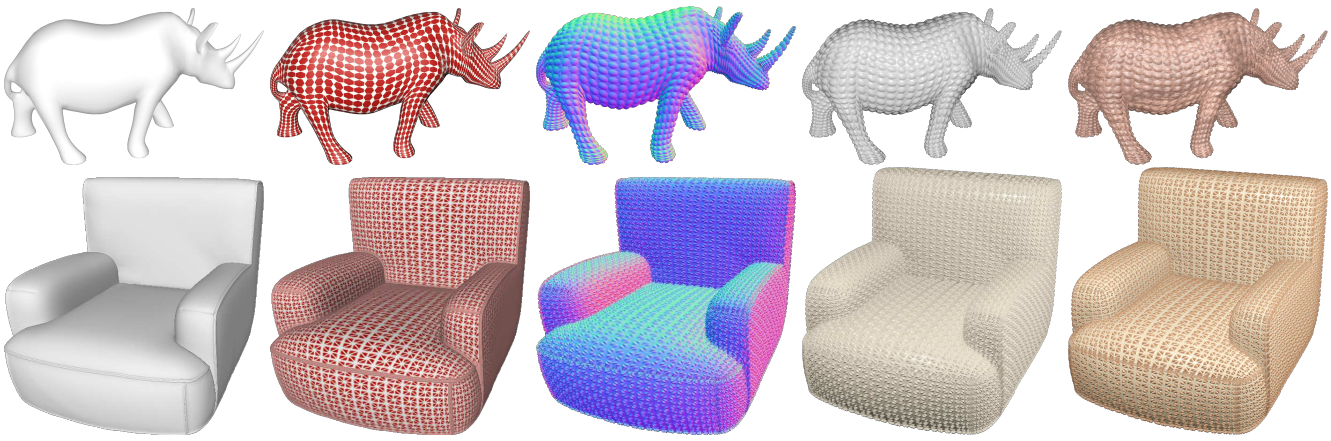
Poisson vector graphics, a powerful extension to the popular diffusion curves, are equipped with two new types of primitives, namely Poisson curves and Poisson regions. This paper has advocated a new decoration approach that functions directly over meshes of arbitrary topological type within the PVG framework. DCs could be utilized to specify color and height on the mesh, serving as the boundary condition. At the same time, both PCs and PRs could control tone by offsetting Laplacians. By using them in concert, we have detailed a pameterization-free decoration method in this paper, which is affording users to directly work on 3D meshes, by sketching out a set of sparse curves intuitively and conveniently enabled by PVG. At the algorithmic level, to render PVG on 3D meshes efficiently, we have designed an efficient Poisson solver based on harmonic B-spline functions in this paper. First, after sampling the knots, we correspondingly built the geodesic Voronoi diagram on the mesh. Second, we used the Voronoi cells to construct a sparse linear system whose solution provides the control coefficients. Finally, we constructed harmonic B-spline functions and locally synthesized the entire result. With the local computation provided by harmonic B-spline functions, we could flexibly and selectively determine which positions of the mesh model are needed for calculation.

In strong contrast to previous works only supported by the utility of DCs, our new system could achieve the decoration results which are embodied not only with the color shading but also with the smooth embossing and carving visual effects. Additionally, because of the smoothness property directly inherited from PRs, our system also support multi-level editing on meshes. Since our system is solely based on a local solver, our system could significantly reduce the computational burden and provide more versatility such as ignoring calculation of invisible parts of models of current interest.

In our current implementation of the PVG solver, we computed Green's functions for mesh vertices only and then use linear interpolation to approximate the function value for points inside triangular faces. As a result, we require the input mesh is of relatively good triangulation and high resolution. It is highly desirable to develop a more accurate method to compute Green's functions for non-vertex points. Also, our paper defines the displacement field using scalar functions. In the future, we plan to extend our method to vector-valued functions so that we can decorate 3D surfaces with non-height function based displacement.

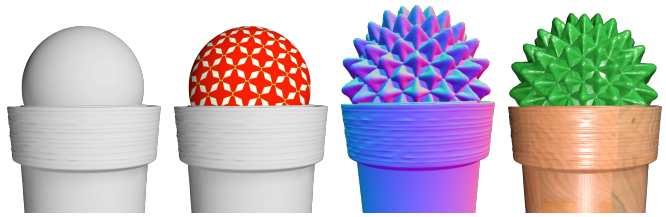
(a) Input Model (b) Displacement constraints (c) Normal map (d) Displacement (e) Color constraints (f) Final result

Figure 9: Decorating the vases with smooth reliefs generated by Poisson regions.

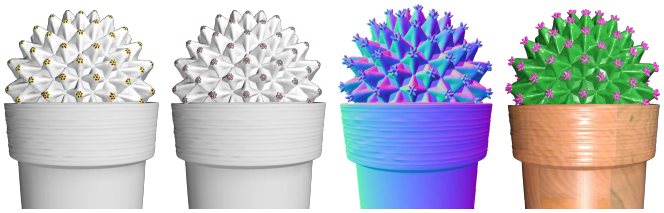


(a) Input mesh (b) Displacement constraints (c) Normal map (d) Result w/o texture (e) Result w/ texture

Figure 10: Decorating the Rhino and Sofa models with repetitive patterns. To simplify the procedure, our system provides global parameterization so that the user-specified pattern can be generated on the entire meshes in a seamless manner.

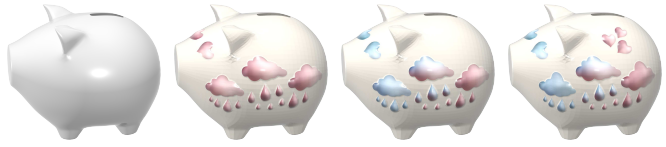


(a) Input model (b) Displacement constraint (c) Normal map (d) Rendering w/ texture

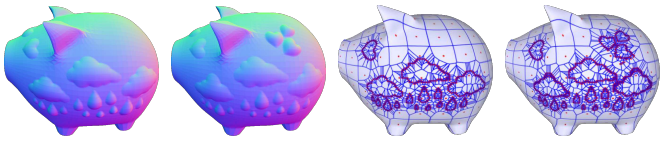


(e) Another displacement constraint (f) Color constraints (g) Normal map (h) Final result

Figure 11: Transforming a sphere into a cactus. We define PVG on 2 separate layers, where the first layer models the stems and the second layer the sharp spines. Since PR is C^1 continuous, the two layers can be added together in a seamless manner. Note that it is difficult to use only DCs for LOD modeling, since it is discontinuous along the curves.

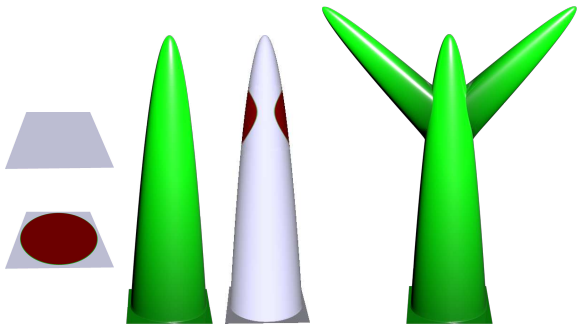


(a) Input model (b) First result (c) Changing color (d) Editing geometry

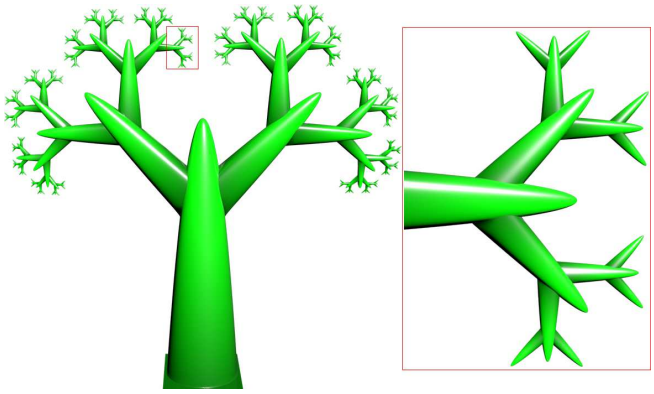


(e) Normal map 1 (f) Normal map 2 (g) First GVD (h) Updated GVD

Figure 12: Decorating and editing a piggy bank model. (a) shows the input mesh. (b) shows the first result of decorating with PVGs. (c) is the second result after changing color whose Voronoi diagram is not changed. (d) is the final result after changing DC's shape of two clouds whose Voronoi diagram is locally updated with some additional knots and only the vertices $\mathbf{x}'$ inside these new Voronoi cells are recalculated by using equation $u(\mathbf{x}') = \sum_{\mathcal{V}_j \in N(\mathbf{x}')} \lambda_j \psi_j(\mathbf{x}')$. There are 1,073 knots in (g), and 214 additional knots are added in (h).



(a) (b) (c) (d)



(e)

Figure 13: Modeling a tree using multilevel PVG. Given a rectangular domain (a), user sketches a circular PR enclosed by a DC whose Dirichlet boundary condition is 0. The resulting PVG is the tree's trunk (b). Then user sketches 2 other sets of DC and PR (c) to define two branches (d). Repeating this procedure a few times, we obtain the tree (e). Similar to other figures, we visualize the values of f and g using colors, i.e., green for 0 and red for positive value.

References

[1] P. V. Sander, H. Hoppe, S. Gortler, J. Snyder, Signal-specialized parameterization (2002) 87–98.

[2] N. A. Carr, J. C. Hart, Painting detail, in: ACM Transactions on Graphics (TOG), Vol. 23, ACM, 2004, pp. 845–852.

[3] M. Stamminger, G. Drettakis, Perspective shadow maps, in: ACM transactions on graphics (TOG), Vol. 21, ACM, 2002, pp. 557–562.

[4] C. Dachsbacher, M. Stamminger, Rendering procedural terrain by geometry image warping, in: Rendering Techniques, 2004, pp. 103–110.

[5] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, D. Salesin, Diffusion curves: A vector representation for smooth-shaded images, ACM Trans. Graph. 27 (3) (2008) 92:1–92:8.

[6] S. Jeschke, D. Cline, P. Wonka, Rendering surface details with diffusion curves, in: ACM SIGGRAPH Asia 2009 Papers, SIGGRAPH Asia '09, ACM, New York, NY, USA, 2009, pp. 117:1–117:8.

[7] F. Hou, Y. He, H. Qin, A. Hao, Knot optimization for biharmonic b-splines on manifold triangle meshes, IEEE Transactions on Visualization and Computer Graphics 23 (9) (2017) 2082–2095.

[8] S. Jeschke, D. Cline, P. Wonka, A GPU Laplacian solver for diffusion curves and Poisson image editing, ACM Trans. Graph. 28 (5) (2009) 116:1–116:8.

[9] S. Boyé, P. Barla, G. Guennebaud, A vectorial solver for free-form vector gradients, ACM Transactions on Graphics (TOG) 31 (6) (2012) 173.

[10] M. Peercy, J. Airey, B. Cabral, Efficient bump mapping hardware, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 303–306.

[11] N. L. Max, Horizon mapping: shadows for bump-mapped surfaces, The Visual Computer 4 (2) (1988) 109–117.

[12] M. J. Kilgard, A practical and robust bump-mapping technique for today's GPUs, in: Game Developers Conference 2000, 2000, pp. 1–39.

[13] T. Kaneko, T. Takahei, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda, S. Tachi, Detailed shape representation with parallax mapping, in: Proceedings of ICAT, Vol. 2001, 2001, pp. 205–208.

[14] N. Tatarchuk, Dynamic parallax occlusion mapping with approximate soft shadows, in: Proceedings of the 2006 symposium on Interactive 3D graphics and games, ACM, 2006, pp. 63–69.

[15] M. M. Oliveira, G. Bishop, D. McAllister, Relief texture mapping, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 359–368.

[16] F. Policarpo, M. M. Oliveira, J. L. Comba, Real-time relief mapping on arbitrary polygonal surfaces, in: Proceedings of the 2005 symposium on Interactive 3D graphics and games, ACM, 2005, pp. 155–162.

[17] R. L. Cook, Shade trees, ACM Siggraph Computer Graphics 18 (3) (1984) 223–231.

[18] R. L. Cook, L. Carpenter, E. Catmull, The reyes image rendering architecture, in: ACM SIGGRAPH Computer Graphics, Vol. 21, ACM, 1987, pp. 95–102.

[19] L. Szirmay-Kalos, T. Umenhoffer, Displacement mapping on the GPU–state of the art, in: Computer Graphics Forum, Vol. 27, Wiley Online Library, 2008, pp. 1567–1592.

[20] G. Elber, Geometric texture modeling, IEEE Computer Graphics and Applications 25 (4) (2005) 66–76.

[21] K. Zhou, X. Huang, X. Wang, Y. Tong, M. Desbrun, B. Guo, H.-Y. Shum, Mesh quilting for geometric texture synthesis, in: ACM Transactions on Graphics (TOG), Vol. 25, ACM, 2006, pp. 690–697.

[22] Y.-K. Lai, S.-M. Hu, D. Gu, R. R. Martin, Geometric texture synthesis and transfer via geometry images, in: Proceedings of the 2005 ACM symposium on Solid and physical modeling, ACM, 2005, pp. 15–26.

[23] P. Bhat, S. Ingram, G. Turk, Geometric texture synthesis by example, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, 2004, pp. 41–44.

[24] R. De Toledo, B. Wang, B. Levy, Geometry textures and applications, in: Computer Graphics Forum, Vol. 27, Wiley Online Library, 2008, pp. 2053–2065.

[25] E. Landreneau, S. Schaefer, Scales and scale-like structures, Comput. Graph. Forum 29 (5) (2010) 1653–1660.

[26] H. Bezerra, E. Eisemann, D. DeCarlo, J. Thollot, Diffusion constraints for vector graphics, in: Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '10, ACM, New York, NY, USA, 2010, pp. 35–42.

[27] M. Finch, J. Snyder, H. Hoppe, Freeform vector graphics with controlled thin-plate splines, ACM Trans. Graph. 30 (6) (2011) 166:1–166:10.

[28] H. Lieng, F. P. Tasse, J. Kosinka, N. A. Dodgson, Shading curves: Vector-based drawing with explicit gradient control, Comput. Graph. Forum 34 (6) (2015) 228–239.

[29] G. Lecot, B. Levy, Ardeco: automatic region detection and conversion, in: 17th Eurographics Symposium on Rendering-EGSR'06, 2006, pp. 349–360.

[30] Y.-K. Lai, S.-M. Hu, R. R. Martin, Automatic and topology-preserving gradient mesh generation for image vectorization, in: ACM Transactions on Graphics (TOG), Vol. 28, ACM, 2009, p. 85.

[31] G. Xie, X. Sun, X. Tong, D. Nowrouzezahrai, Hierarchical diffusion curves for accurate automatic image vectorization, ACM Transactions on Graphics (TOG) 33 (6) (2014) 230.

[32] P. Feng, J. Warren, Discrete bi-laplacians and biharmonic b-splines, ACM Trans. Graph. 31 (4) (2012) 115:1–115:11.

[33] F. Hou, Q. Sun, Z. Fang, Y.-J. Liu, S.-M. Hu, H. Qin, A. Hao, Y. He, Poisson vector graphics (pvg) and its closed-form solver, arXiv:1701.04303.

[34] Y. Lipman, R. M. Rustamov, T. A. Funkhouser, Biharmonic distance, ACM Transactions on Graphics (TOG) 29 (3) (2010) 27.

[35] E. Nava-Yazdani, K. Polthier, De casteljau's algorithm on manifolds, Computer Aided Geometric Design 30 (7) (2013) 722–732.

[36] S.-Q. Xin, X. Ying, Y. He, Constant-time all-pairs geodesic distance query on triangle meshes, in: Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games, ACM, 2012, pp. 31–38.

[37] S.-Q. Xin, G.-J. Wang, Improving chen and han's algorithm on the discrete geodesic problem, ACM Trans. Graph. 28 (4) (2009) 104:1–104:8.

[38] Y.-J. Liu, Z. Chen, K. Tang, Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1502–1517.

[39] S. Kumar, D. Manocha, W. Garrett, M. Lin, Hierarchical back-face computation, Computers & Graphics 23 (5) (1999) 681 – 692.

[40] N. Greene, M. Kass, G. Miller, Hierarchical z-buffer visibility, in: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, ACM, New York, NY, USA, 1993, pp. 231–238.