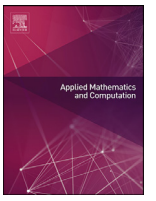
Applied Mathematics and Computation 451 (2023) 128031

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc



Geometric texture transfer via local geometric descriptors

Martin Huska[a], Serena Morigi[a,*], Giuseppe Recupero[a]

Department of Mathematics, University of Bologna, Bologna, Italy

ARTICLE INFO

Article history:
Received 14 November 2022
Revised 1 March 2023
Accepted 2 April 2023
Available online 21 April 2023

Keywords:
Geometric texture transfer
Differential descriptors
Nonlinear least squares
Variational formulation

ABSTRACT

Geometric Texture Transfer, aimed to add fine grained details to surfaces, can be seen as a realistic advanced geometry modelling technique. At this aim, we investigate and advocate the use of local geometric descriptors as alternative descriptors to the vertex coordinates for surface representation. In particular, we consider the Laplacian coordinates, the normal-controlled coordinates and the mean value encoding, which are well prone to facilitate the transfer of source geometric texture details onto a target surface while preserving the underlying global shape of the target surface. These representations, in general, encode the underlying geometry by describing relative position of a vertex with respect to its local neighborhood, with different levels of invariance to rigid transformations and uniform scaling. We formulate the geometric texture transfer task as a constrained variational nonlinear optimization model that combines an energy term on the shape-from-operator inverse model with constraints aimed to preserve the original underlying surface shape. In contrast to other existing methods, which rely on the strong assumptions of bijectivity, equivalency in local connectivity, and require massive tesselations, we simply map the geometric texture on the base surface, under the only assumption of boundary matching. The proposed geometric texture transfer optimization model is then efficiently solved by nonlinear least squares numerical methods. Experimental results show how the nonlinear texture transfer variational approach based on mean value coordinates overcomes the performance of other alternative descriptors.

1. Introduction

Nowadays extensively used in computer graphics, the bump mapping technique makes the base surface of an object (macrostructure) appears wrinkled, wavy, embossed, [1]. This effect of bumps on surfaces is an apparent variation of the local geometry due to a variation of the normal in a point. The geometric normal of the object remains unchanged and so the geometry of the object, the only variation concerns the normal vectors used in the local lighting model that produces a variation in shading and thus rendering of the surface.

Geometric Texture Transfer (GTT), also named displacement mapping, in contrast, uses a height-field texture function to perturb points on a base surface along the direction of the normals to the surface. Small perturbation actually changes the geometry of the model instead of modulating only the parameters of shading as the bump mapping technique. This allows us to classify GTT as a realistic advanced mesh modeling technique: geometric details are mapped/transferred onto a base surface defined in general by a coarse mesh. The level of details is represented by the geometric texture, a small scale

* Corresponding author.
E-mail addresses: martin.huska@unibo.it (M. Huska), serena.morigi@unibo.it (S. Morigi), giuseppe.recupero3@unibo.it (G. Recupero).

https://doi.org/10.1016/j.amc.2023.128031

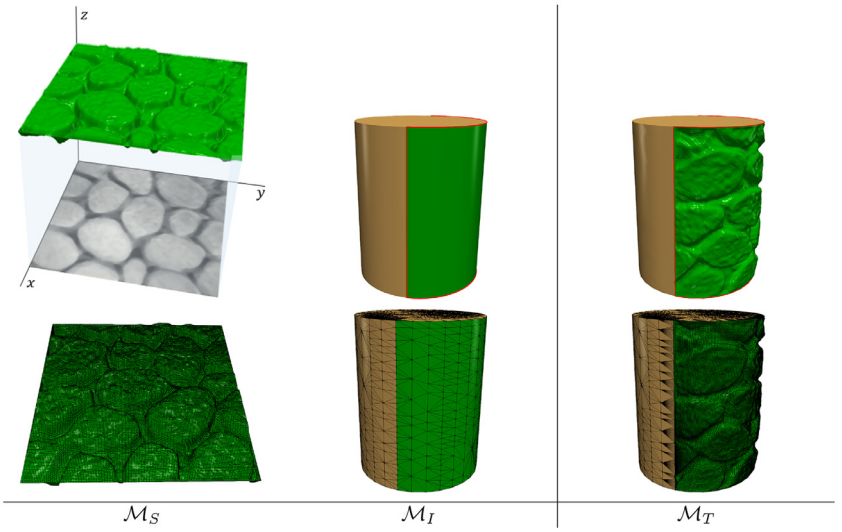


Fig. 1. The geometric transfer process. First column: geometric texture $\mathcal{M}_S$, derived from the gray-scale texture image; second column: base mesh $\mathcal{M}_I$, with the patch $P \subset \mathcal{M}_I$ colored in green; third column: textured mesh $\mathcal{M}_T$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

deformation function locally affecting the surface patch. In standard displacement mapping the actual geometric position of points over the patch is displaced, along the local surface normal, according to the value of the texture function. A broader meaning of GTT also includes geometry textures represented by 3D meshes, which leads to a more sophisticated transfer task.

To make the transfer of geometric details as effective as possible, it is first of all essential to represent them at their best. The Euclidean coordinates have been a ubiquitous choice in geometry processing due to their intuitive properties of representing the spatial embedding of a shape, however they fail to capture the geometric features relevant for many shape analysis tasks [2]. To address this challenge, several alternative descriptors for shape analysis have been proposed [3,4]. They naturally encode aspects of extrinsic geometry, and therefore they are potentially suitable also for the GTT task. In this work we analyse the use of geometric descriptors such as the differential coordinates [3], the normal-controlled coordinates [4], and the mean value encoding [5], as alternative representations for the Euclidean vertex coordinates. These representations will allow to better code the high frequency details of the geometric texture, and, consequently, to preserve them during the transfer onto another surface. In particular, we investigate their properties of invariance under affine transformations and free deformations. This represents indeed a key aspect towards their use in the context of GTT task.

Texturing a given surface has a natural formulation for parameterized surfaces, based on UV-mapping, following the theory of classical displacement mapping [6]. Let us denote the mesostructure surface $\mathcal{M}_T$ by the parametric form $M_T(u,v) : \mathbb{R}\times\mathbb{R} \to \mathbb{R}^3$, the macrostructure base surface $\mathcal{M}_I$ by $M_I(u,v) : \mathbb{R}\times\mathbb{R} \to \mathbb{R}^3$, and the displacement (geometric texture) by a height map function $M_S(u,v) : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$. The height map is a gray scale texture image in the texture coordinates $u, v \in [0,1]^2$. Displacement mapping allows to obtain a mesostructure $\mathcal{M}_T$ by mapping on a macrostructure base surface $\mathcal{M}_I$ the geometric texture $\mathcal{M}_S$ in the direction of the macrostructure normal vector $N_I(u,v)$. Assuming that both $\mathcal{M}_I$ and $\mathcal{M}_S$ share a common parameterization domain, the displacement mapping reads as

$$M_T(u,v) = M_I(u,v) + N_I(u,v)M_S(u,v). \tag{1}$$

However, in most real applications, 3D shapes are commonly represented as arbitrary topology, irregular, triangular meshes, which demand for texture atlases generation and charts parameterization, before applying the direct formulation (1). To face this challenge in our GTT proposal the macrostructure base surface is a Riemannian manifold of arbitrary topology embedded in $\mathbb{R}^3$ represented by an irregular *base mesh* $\mathcal{M}_I$, defined by its vertices, edges and triangular faces (V_I, E_I, T_I), respectively, and the displacement/*geometric texture* is an open mesh $\mathcal{M}_S = (V_S, E_S, T_S)$ with boundary $b_S := \partial\mathcal{M}_S$ defined by a geometry image. We assume that the underlying base mesh is coarse relative to the fine scale details of the texture. We aim at building a new *textured mesh* $\mathcal{M}_T = (V_T, E_T, T_T)$, which corresponds to the base mesh $\mathcal{M}_I$ everywhere except in a bounded patch $P \subset \mathcal{M}_I$ where the geometric texture is mapped onto the base mesh, while preserving as much as possible the underlying original shape of $\mathcal{M}_I$, i.e. $V_T|_{\mathcal{M}_I - P} = V_I|_{\mathcal{M}_I - P}$, and $V_T|_P$ is displaced by transferring the level of details of V_S.

The geometric transfer process is illustrated in Fig. 1 where a column of stones is modelled by mapping on one half of a cylindrical shape a geometric textured surface built from a gray-scale image. The involved underlying meshes are represented in Fig. 1, bottom row.

The main contributions of this work are the following:

- we investigate numerical aspects of both the direct and the inverse representation problem (derive the Euclidean geometry given the descriptors of a shape) using geometric descriptors as Laplacian, normal-controlled and mean value coordinates, highlighting their key properties under transformations as well as the numerical solution to the inverse ill-posed shape-from-operator problem;
- associated with the three considered geometric descriptors, we formulate and analyse three variational models for the solution of the GTT problem which involve two terms: one for the inverse reconstruction problem and one for shape preserving interpolation;
- we design a nonlinear numerical optimisation algorithm to efficiently solve the GTT models and we compare the results of transferring geometric textures both qualitatively and quantitatively on a wide range of examples.

The paper is organized as follows. In Section 2 we briefly review the approaches for the solution of the GTT task. Section 3 introduces the local shape descriptors considered, and their invariance under affine transformations is analysed in Section 4. The variational models proposed for GTT are described in Section 5. Sections 6 describes in detail the main ingredients for the numerical solution of the optimisation methods designed to solve the GTT problem. In Section 7 the proposed methods are validated on several types of surfaces and geometric textures. Finally, in Section 8 some conclusions are reported.

2. Related works in geometric texture transferring

GTT, originally named displacement mapping, was introduced in the field of computer graphics as the 3-dimensional extension of traditional image-based texture mapping [7]. While texture mapping assigns color to surface points, displacement mapping assigns vector offsets to create a variety of geometric details. Blinn [1] proposed perturbing surface normals using a wrinkle function to address specific computer graphics visualization purposes. Recently introduced hardware tesselation units gave a new potential to GPU displacement mapping [8]. In [9] the GPU GTT proposal relies on subdivision limit base surfaces and spline displacement functions.

Besides the specific visualization purpose, GTT has later been applied in geometric modeling to enrich a 3D model with a variety of geometric details, without the effort needed to manually specify them.

Classical methods dealing with GTT differ in the mathematical form used to represent both the base surface and the geometric texture, which consequently affects the numerical approaches used in the transfer task.

The majority of the proposals are based on *polygonal mesh representations*, relying on a consistent surface parameterization between source and target shape. The use of geometric descriptors and shared parameterization for the transferring of geometric details has been preliminarly presented in [3] using the Laplacian coordinates and more recently in [10] via mean curvature details. Another GTT approach was described in [11], using a traditional texture atlas approach, where both the base surface and the texture patch are preliminarly mapped onto geometry images. A natural extension from texture image was presented in [12] where the displacement map is described via trivariate functions over a volume parametric form. To address the critical parameterization issue, in [13] the authors combine *implicit representation* of the underlying smooth surface with so-called detail particles, i.e. particle sets containing the offset vectors from smoothed surface. The texture transfer task consists in level-set evolution driven by speed function derived from the detail particles.

In [14] GTT is based on *spectral shape representation*. The authors apply a multi-scale empirical mode decomposition (EMD) to the source object and the GTT task is performed by adding selected modes to the EMD of the target surface. A similar spectral-based idea is presented in [15] where the geometric texture from source object is encoded via spectral decomposition of Laplacian, then combined with the spectral decomposition of the target.

With the rapid development of machine learning approaches, new *neural implicit representations* for 3D geometry have been introduced. In [16,17] the authors applied a convolutional neural network to synthesize a geometric texture from sample object and to transfer it over whole target object. GTT using neural representations is memory/computation demanding and still very challenging in transferring texture on a specific bounded patch of a surface.

In this work, we focus on polygonal mesh representations, which guarantee an accurate explicit geometric texture representation, essential for geometric modelling purposes. In contrast to the previous approaches we avoid the use of a global parameterization $(u, v, atlas)$, and we do not require any mesh refinement of the base surface and texture to adapt one to the other. Even if we mainly handle height-field geometric textures, we detail and showcase the extension to generic 3D texture sample meshes, which is a well-known critical issue in case of GTT with mesh representation. With respect to GTT using implicit representations, the proposed GTT approaches avoid the timing and accuracy shortcomings of the level set modeling. Finally, unlike the multi-block procedures proposed in computer graphics literature to deal with the GTT task, we face the problem from a numerical point of view, offering a simple, compact mathematical variational formulation with efficient solutions.

The standard encoding of discrete geometric models has always relied on Euclidean coordinates of the vertices defined with respect to a common global coordinate system. However, several alternative representations have been proposed as better performing solutions for specific processing tasks. The Laplacian coordinates (LAP) [3,18] have been introduced in the context of mesh editing, the normal-controlled coordinates (NCC) [4] have been successfully applied to mesh deformation and mean value coordinates (MVE) [5] have been proposed for morphing and blending. Spectral-based encoding methods extract geometric information directly from operator eigenvalues and eigenfunctions [19,20] and naturally separate features

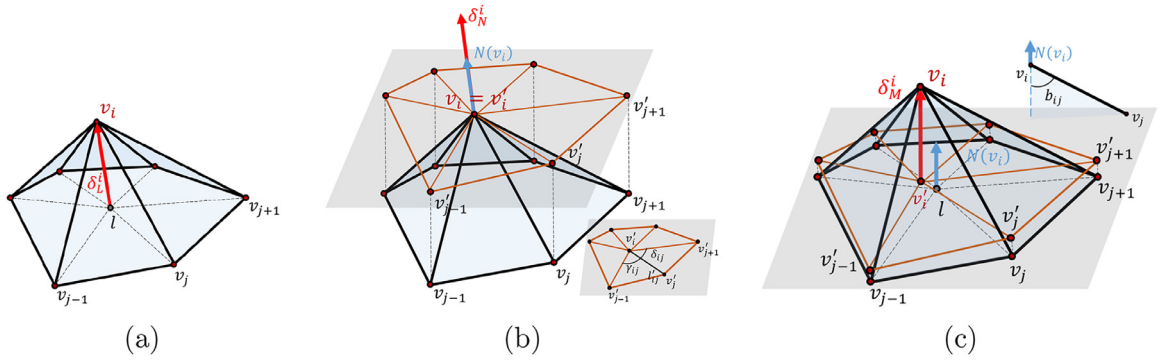


Fig. 2. Local shape descriptors construction: (a) δ_L^i, (b) δ_N^i, with projection plane, (c) δ_M^i, with cotangent b_{ij}.

at different level of scale. They have been successfully used in mesh segmentation [20,21], mesh compression [22], and other relevant mesh processing tasks [19]. However, they suffer from the high computational cost. In [23] the authors proposed the height-and-tilt texture encoding for geometric texture editing and animation tasks.

The proposed GTT variational approach is a general framework and we compare, in particular, the most used local geometric descriptors such as LAP [3], NCC [4] and MVE [5], which easily adapt to the change of the global shape while respecting the structural detail as much as possible. Efficient schemes will be discussed for their direct and shape-from-operator inverse evaluation.

3. Represent a mesh by local shape descriptors

Many geometry processing tasks, like editing, morphing, deformation, blending and, in particular, GTT, benefit enormously from using alternative intrinsic mesh representations, able to encode the shape details of the mesh and easier to adapt to the considered tasks.

We briefly resume the main alternative local shape descriptors for mesh representations, how these descriptors can be computed on a given mesh $\mathcal{M} = (V, E, T)$, which is denoted as direct problem, and how to solve the inverse shape-from-operator problem, i.e. how to recover information about the mesh structures, in particular the vertex locations, from the operators and the descriptors. The shape-from-operator inverse problem is ill-posed as the recovery of the vertices can be obtained up to isometric transformations and involves non-invertible operators. Finally in Section 4, we discuss their limits in being completely affine-invariant representations.

3.1. Differential coordinates

The simplest form of differential coordinates is represented by the Laplacian coordinates, here named *LAP*, which rely on the discretization of the continuous linear Laplacian operator at vertex v_i. A mesh Laplacian operator locally takes the difference between the value of a function at a vertex v_i and a weighted average of its values at the first-order neighbour vertices:

$$L(v_i) = v_i - \sum_{j\in\mathcal{N}(v_i)} w_{ij} v_j = \sum_{j\in\mathcal{N}(v_i)} w_{ij}(v_i - v_j), \tag{2}$$

where $\mathcal{N}(v_i)$ represents the first ring of neighbors of v_i, and $w_{ij} > 0$ are the weights normalized as $w_{ij} = \bar{w}_{ij} / \sum_{j\in\mathcal{N}(v_i)} \bar{w}_{ij}$, so that $\sum_{j\in\mathcal{N}(v_i)} w_{ij} = 1$. Common choices for w_{ij} are the uniform weights, i.e. $w_{ij} = 1/d_i$, which, depending only on the valence d_i of each vertex, do not explicitly encode geometric information, and the cotangent weights [24] which more faithfully represent discretizations of the Laplace-Beltrami operator from Riemannian geometry.

The LAP coordinates associated to vertex v_i are represented by the displacement vector $L(v_i) \in \mathbb{R}^3$ in (2) denoted as $\delta_L^i = (\delta_L^{(x_i)}, \delta_L^{(y_i)}, \delta_L^{(z_i)})$, illustrated in Fig. 2(a). Given the Euclidean coordinates $V = (V^{(x)}, V^{(y)}, V^{(z)})$ of n_V vertices, the LAP-coords $\delta_L = (\delta_L^{(x)}, \delta_L^{(y)}, \delta_L^{(z)}) \in \mathbb{R}^{n_V \times 3}$ are obtained by the following linear mapping

$$\delta_L = L_w V, \tag{3}$$

where the linear operator in (2) is represented in (3) by a sparse matrix $L_w \in \mathbb{R}^{n_V \times n_V}$ with components defined as:

$$(L_w)_{ij} = \begin{cases} 1 & if \quad i = j \\ -w_{ij} & if \quad (i,j) \in E \\ 0 & otherwise. \end{cases} \tag{4}$$

The fact that the Laplacian coordinates allow to represent local details at each surface vertex better than the Euclidean coordinates derives from the well known relation $\Delta_{\mathcal{M}} V = 2\mathcal{H}$, see [24], discretized at i-th vertex as $(L_w V)_i \approx 2 n_i H_i$, with n_i

the normal to the vertex and H_i its associated local mean curvature. Thus the direction of δ_L approximates the normal, and the magnitude approximates the mean curvature.

Under a graph-based view of a mesh, given a mesh, represented by the underlying graph $\mathcal{G}$, the graph Laplacian of $\mathcal{G}$ is related to the incidence matrix B of $\mathcal{G}$ as $L_w = BB^T$. The rank of the Laplacian matrix L_w is equal to the rank of the symmetric incidence matrix B for $\mathcal{G}$, i.e. $rank(L) = rank(B) = n - K$, where n is the number of vertices and K the number of connected components. It follows that the matrix L_w is not full rank even for an open connected mesh, and, consequently, the operator in (4) is not invertible. Therefore, given the δ_L coordinates, the inverse process recovers the set of vertices V by solving the linear least squares optimization problem

$$V^* \in \arg\min_V \|L_w V - \delta_L\|_2^2, \tag{5}$$

which has an infinite number of solutions. A unique reconstruction can be obtained by fixing at least one vertex position for a connected mesh.

3.2. Normal-controlled coordinates

Introduced in [4] the Normal-Controlled Coordinates (NCC) represent a generalization of the Laplacian coordinates. Fig. 2(b) illustrates the construction of these coordinates. The normal to the vertex v_i defines an orthogonal plane where the neighbor vertices are projected, see Fig. 2(b), bottom part. Then the local parameterization of a vertex $v_i \in V$ is defined with respect to the projected vertices v'_j with associated weights

$$\bar{w}_{ij} = \frac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{\|v_i - v'_j\|}. \tag{6}$$

The weights in (6), called mean-value coordinates [25], involve the angles formed by the edges of the first ring of the vertex, in the local projected plane. Then, for each vertex v_i the associated NCC is a vector in $\mathbb{R}^3$, illustrated in Fig. 2(b), describing the local geometry feature at v_i, which is always parallel to the vertex normal, independently by its definition.

Given the Euclidean coordinates $V = (V^{(x)}, V^{(y)}, V^{(z)})$, the associated NCC, denoted by $\delta_N = (\delta_N^{(x)}, \delta_N^{(y)}, \delta_N^{(z)})$, are obtained by the linear mapping

$$\delta_N = N_w V, \tag{7}$$

where the weight matrix N_w is sparse, non-symmetric with elements:

$$(N_w)_{ij} = \begin{cases} 1 & if \quad i = j \\ -w_{ij} & if \quad (i,j) \in E \\ 0 & otherwise, \end{cases} \tag{8}$$

and w_{ij} are the normalized weights $w_{ij} = \bar{w}_{ij} / \sum_{j \in \mathcal{N}(v_i)} \bar{w}_{ij}$.

An important property of the matrix L_w, imposed by equation (2), which similarly holds also for N_w, is the zero row sum. If V is a constant vector, i.e., a vector whose components are the same, then V lies in the kernel of L_w, since $L_w V = 0$ for an operator L_w with zero row sum. This implies that the constant vectors are eigenvectors of L_w with eigenvalue zero. As a result, $dim(ker(N_w)) \geq 1$, hence L_w and N_w are not invertible.

Therefore, given the δ_N coordinates, the inverse process recovers a set of coordinates V by solving the linear least squares minimization problem

$$V^* \in \arg\min_V \|N_w V - \delta_N\|_2^2, \tag{9}$$

which, by imposing suitable Dirichlet boundary conditions, has a unique solution.

3.3. Mean value encoding

Introduced for planar triangulation in [25], and then generalized in $\mathbb{R}^3$ [5], the Mean Value Encoding (MVE) coordinates represent an improvement of the Pyramid Coordinates [26]. The construction of the MVE coordinates is illustrated in Fig. 2(c). For each vertex $v_i \in V$, an approximated normal n_i is computed as

$$n_i = \frac{\sum_{j=1}^{|\mathcal{N}(v_i)|} (v_j - l) \times (v_{j+1} - l)}{\| \sum_{j=1}^{|\mathcal{N}(v_i)|} (v_j - l) \times (v_{j+1} - l) \|}, \qquad l = \frac{1}{|\mathcal{N}(v_i)|} \sum_{j \in \mathcal{N}(v_i)} v_j. \tag{10}$$

Then the associated MVE representation is uniquely obtained by first computing the set of coefficients (w_{ij}, b_{ij}), $j \in \mathcal{N}(v_i)$, defined for each half-edge $w_{ij} \neq w_{ji}$, $b_{ij} \neq b_{ji}$ as follows

$$w_{ij} = \bar{w}_{ij} / \textstyle\sum_{j \in \mathcal{N}(v_i)} \bar{w}_{ij}, \quad \bar{w}_{ij} = \dfrac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{\|v'_i - v'_j\|} \tag{11}$$

$$b_{ij} = \frac{c_{ij}}{\sqrt{1 - c_{ij}^2}}, \qquad c_{ij} = \frac{(v_i - v_j) \cdot n_i}{\|v_i - v_j\|}. \tag{12}$$

T_g	L_w	δ_L	N_w	δ_N	w	b	δ_M
Translation	✓	✓	✓	✓	✓	✓	✓
Rotation	✓	X	✓	X	✓	✓	X
Isotropic Scaling	✓	X	✓	X	✓	✓	X

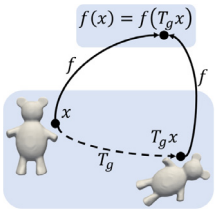


Fig. 3. Geometric descriptor G-invariance $f(T_g x) = f(x)$. Left: Table for all descriptors. Right: Illustration of G-invariance map f.

Given the Euclidean coordinates $V = (V^{(x)}, V^{(y)}, V^{(z)})$, the associated MVE-coords $\delta_M = (\delta_M^{(x)}, \delta_M^{(y)}, \delta_M^{(z)})$ are obtained by the following nonlinear mapping

$$\delta_M = \mathcal{V}(V), \tag{13}$$

where $\forall v_i \in V$

$$\mathcal{V}(v_i) = \sum_{j \in \mathcal{N}(v_i)} w_{ij}(\|v_i' - v_j'\| b_{ij} + (v_j - v_j') \cdot n_i) n_i.$$

Note that δ_M represents the displacement of v_i from v_i', its projection onto the local projection plane orthogonal to n_i.

Given the δ_M coordinates, the inverse process recovers the Euclidean coordinates of $v_i \in V$ by the following closed-form expression as function of the rest of the neighborhood vertices

$$v_i = v_i' + \delta_M = F_i(V; w, b) = \sum_{j \in \mathcal{N}(v_i)} w_{ij} \left(v_j + \|N_i \sum_{k \in \mathcal{N}(v_i)} w_{ik}(v_k - v_j)\| b_{ij} n_i \right), \tag{14}$$

with $N_i = I_3 - n_i n_i^T$. Note that $F_i(V; w, b)$ has not direct dependence on v_i, but only on the first-ring vertices of v_i and on the normal vector n_i at v_i. As mentioned in [5], MVE coordinates depend continuously on the data, thus small variations of vertices in $\mathcal{N}(v_i)$ result in a small variation of v_i.

Given the MVE coordinates, the inverse process recovers V by solving the nonlinear least squares minimization problem

$$V^* \in \arg\min_V \|V - F(V)\|_2^2 = \sum_i \|v_i - F_i(V; w, b)\|_2^2, \tag{15}$$

with the nonlinear function $F_i(V)$ defined in (14), and the components of the vectors w and b defined in (11) and (12), respectively.

4. Invariance of shape descriptors

Let G be the affine transformation group (containing translations, rotations, shearing and scaling). A n-dimensional real representation of a group G is a map $T_g : G \to \mathbb{R}^{n \times n}$, assigning to each $g \in G$ an invertible matrix T_g. We are interested in transformations in homogeneous space $T_g \in \mathbb{R}^{4 \times 4}$ represented as

$$T_g = \begin{bmatrix} a_1 & b_1 & c_1 & t_x \\ a_2 & b_2 & c_2 & t_y \\ a_3 & b_3 & c_3 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{16}$$

where the principal three-by-three sub-matrix represents linear transformations such as scaling and rotation, while $t = [t_x, t_y, t_z]^T$ denotes the translation vector. To characterize the invariance properties of the shape descriptors, we introduce the following concepts of invariance with respect to transformations $g \in G$.

Let Ω be the domain underlying our data, and $X(\Omega)$ the space of functions on Ω. A function $f : X(\Omega) \to Y$ is *G-invariant* if

$$f(T_g x) = f(x), \tag{17}$$

for all $g \in G$, and $x \in X(\Omega)$, i.e., its output is unaffected by the group action on the input.

A function $f : X(\Omega) \to X(\Omega)$ is *G-equivariant* if

$$f(T_g x) = T_g f(x), \tag{18}$$

for all $g \in G$, i.e., group action on the input affects the output in the same way.

We focus now on the main transformations involved in the geometric transfer task, such as translation, rotation along an axis, and isotropic scaling, represented as T_g in (16). In Fig. 3, left, and Fig. 4, left, we summarize the properties of G-invariance and G-equivariance of the shape descriptors presented in Section 3; with the symbol ✓ we denote satisfied, and

T_g	δ_L	δ_N	δ_M
Translation	X	X	X
Rotation	✓	✓	✓
Isotropic Scaling	✓	✓	✓

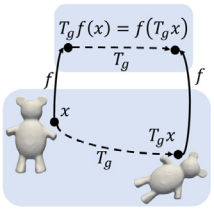


Fig. 4. Geometric descriptor G-equivariance $f(T_g x) = T_g f(x)$. Left: Table for all descriptors. Right: Illustration of G-equivariance map f.

by X not satisfied. The difference between the concepts of G-invariance and G-equivariance maps is illustrated in the right panels of Fig. 3 and Fig. 4, respectively.

Laplacian coordinates. A transformation T_g changes the geometry of the mesh, leaving unchanged its local connectivity, thus the uniform weights w_{ij} in (2) remain invariant and not equivariant. In case the cotangent weights w_{ij} are used, they remain invariant since the angles in the local neighborhood are neither modified by rigid transformation, nor by isotropic scaling. Therefore, the Laplacian L_w is an intrinsic linear operator which remains invariant under isometric transformations - such as translations and rotations in $\mathbb{R}^3$ - which do not change the metric. A different behaviour is expected for the LAP coordinates, as shown in Prop. 4.1.

Proposition 4.1. *Let δ_L be the map defined in (3). Then δ_L is G-invariant for T_g translation, and G-equivariant for T_g being a rotation along an axis or an isotropic scaling.*

Proof. Let δ_L be the vector of LAP coords of the mesh $\delta_L = L_w(V)$ and δ_L' the vector of the LAP coords of the transformed mesh $\delta_L' = L_w(T_g V)$. If $T_g = T$, with T the translation matrix obtained by replacing in T_g in (16) the principal submatrix with the I_3 identity matrix. Then

$$(\delta_L')_i = (L_w(TV))_i = \sum_j w_{ij}(v_i + t - v_j - t) = (L_w(V))_i = (\delta_L)_i.$$

If $T_g = S$, with S the diagonal matrix obtained by setting $a_1 = b_2 = c_3 = s \in \mathbb{R}$ in (16), which represents the isotropic scaling of a factor s. Then, due to the linearity of L_w,

$$(\delta_L')_i = (L_w(SV))_i = \sum_j w_{ij}(sv_i - sv_j) = (sL_w(V))_i = (S\delta_L)_i.$$

If $T_g = \mathcal{R}$, with $\mathcal{R}$ an orthogonal rotation matrix, then

$$(\delta_L')_i = (L_w(\mathcal{R}V))_i = \mathcal{R}v_i - \sum_j w_{ij}\mathcal{R}v_j = \mathcal{R}\left(v_i - \sum_j w_{ij}v_j\right) = \mathcal{R}(L_w(V))_i = (\mathcal{R}\delta_L)_i.$$

□

Normal-controlled coordinates. Similarly to the Laplacian matrix L_w, the weights w_{ij} of N_w in (6) remain unchanged under translation and rotation, which do not affect the shape of the projected neighborhood. Isotropic scaling does not alter γ_{ij} and δ_{ij} in (6), while affects l_{ij}. Nevertheless, the weight normalization $w_{i,j}$ from $\bar{w}_{i,j}$ in (6) allows to preserve the invariance under isotropic scaling. Following the same arguments of Prop. 4.1, one can show that the map δ_N defined in (7) is invariant for translation and equivariant for rotation and scaling.

Mean Value Encoding. The weights w_{ij}, defined in (11), differ from NCC weights only for the choice of the local projection plane. Thus the w_{ij} are invariant under all the affine transformations. The components b_{ij}, as illustrated in Fig. 2(c), define the angle between the edge e_{ij} and the vertex normal n_i, which remains preserved both for rigid transformation and isotropic scaling. The coordinates δ_M combine non-linearly the invariant weights w_{ij}, b_{ij} with the normal vector n_i, thus the invariance/equivariance of δ_M is less trivial to verify.

Proposition 4.2. *Let δ_M be the map defined in (13). Then δ_M is G-invariant for T_g translation, and G-equivariant for T_g being a rotation along an axis or an isotropic scaling.*

Let F be the map defined in (14). Then F is G-invariant for T_g translation, rotation and uniform scaling.

Proof. Let us define the average distance from origin as

$$d(v_i) := -\frac{1}{|\mathcal{N}(v_i)|}\sum_{j\in\mathcal{N}(v_i)}\langle n_i, v_j\rangle.$$

For T_g defining translation, for each vertex $T_g v_i = v_i + t$ for $t = [t_x, t_y, t_z]^T \in \mathbb{R}^3$ and $d(v_i + t) = d(v_i) - \langle n_i, t\rangle$. Then

$$(v_i + t)' = v_i + t - (d(v_i) - \langle n_i, t\rangle + \langle v_i + t, n_i\rangle)n_i = v_i' + t. \tag{19}$$

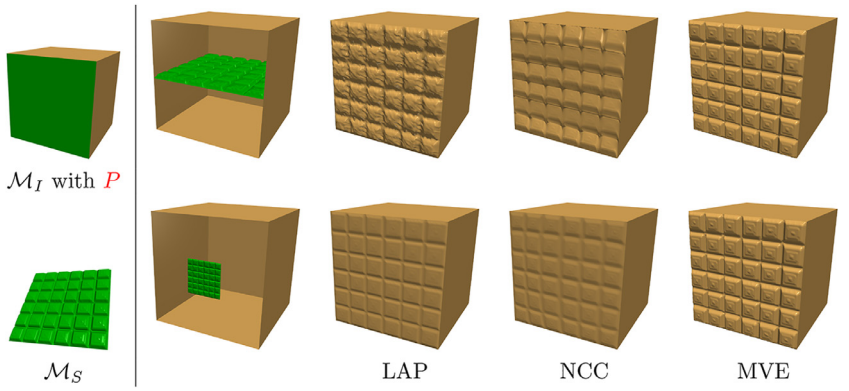


Fig. 5. Invariance w.r.t GTT task. Left: GTT set up. Right: placing $\mathcal{M}_S$ with respect to P by rotation (first row) and by isotropic scaling (second row)..

By applying (19), we get

$$\mathcal{V}(T_g v_i) = \mathcal{V}(v_i + t) = \sum_{j\in\mathcal{N}(v_i)} w_{ij}(\|((v_i+t)' - (v_j+t)')\|b_{ij} + (v_j + t - (v_j+t)')\cdot n_i)n_i =$$
$$= \sum_{j\in\mathcal{N}(v_i)} w_{ij}(\|(v_i' - v_j')\|b_{ij} + (v_j - v_j')\cdot n_i)n_i = \mathcal{V}(v_i) = \delta_M.$$

For T_g being rotation, due to the orthogonality of T_g and the fact that $n_i(T_g v_i) = T_g n_i$, we have

$$d(T_g v_i) = -\frac{1}{|\mathcal{N}(v_i)|}\sum_{j\in\mathcal{N}(v_i)} \langle T_g n_i, T_g v_j\rangle = -\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(v_i)} n_i^T T_g^T T_g v_j = d(v_i)$$

and the projection of rotated vertices is given by

$$(T_g v_i)' = T_g v_i - (d(v_i) + \langle T_g v_i, T_g n_i\rangle)T_g n_i = T_g(v_i - (d(v_i) + \langle v_i, n_i\rangle)n_i = T_g v_i'.$$

Then

$$\mathcal{V}(T_g v_i) = \sum_{j\in\mathcal{N}(v_i)} w_{ij}\Big(\|T_g v_i' - T_g v_j'\|b_{ij} + \big\langle T_g v_j - T_g v_j', T_g n_i\big\rangle\Big)T_g n_i =$$
$$= \sum_{j\in\mathcal{N}(v_i)} w_{ij}\Big(\|T_g(v_i' - v_j')\|b_{ij} + \big\langle T_g(v_j - v_j'), T_g n_i\big\rangle\Big)T_g n_i =$$
$$= T_g\left(\sum_{j\in\mathcal{N}(v_i)} w_{ij}\Big(\|v_i' - v_j'\|b_{ij} + \big\langle v_j - v_j', n_i\big\rangle\Big)n_i\right) = T_g\,\mathcal{V}(v_i).$$

The isotropic scaling can be expressed as $T_g = sI_3$, where $s = a_1 = b_2 = c_3 \in \mathbb{R}$ in T_g, and it affects the normal vector n_i. However,

$$d(T_g v_i) = -\frac{1}{|\mathcal{N}(v_i)|}\sum_{j\in\mathcal{N}(v_i)} \langle n_i, T_g v_j\rangle = -\frac{1}{|\mathcal{N}(v_i)|}\sum_{j\in\mathcal{N}(v_i)} n_i^T(sI_3)v_j = sd(v_i)$$

and

$$(T_g v_i)' = T_g v_i - (sd(v_i) + \langle T_g v_i, n_i\rangle)n_i = T_g(v_i - (d(v_i) + \langle v_i, n_i\rangle)n_i = T_g v_i'.$$

Therefore

$$\mathcal{V}(T_g v_i) = \sum_{j\in\mathcal{N}(v_i)} w_{ij}\Big(\|T_g(v_i' - v_j')\|b_{ij} + \big\langle T_g(v_j - v_j'), n_i\big\rangle\Big)n_i =$$
$$= \sum_{j\in\mathcal{N}(v_i)} w_{ij}\Big(s\|v_i' - v_j'\|b_{ij} + s\big\langle v_j - v_j', n_i\big\rangle\Big)n_i$$
$$= T_g\left(\sum_{j\in\mathcal{N}(v_i)} w_{ij}\Big(\|v_i' - v_j'\|b_{ij} + \big\langle v_j - v_j', n_i\big\rangle\Big)n_i\right) = T_g\,\mathcal{V}(v_i).$$

□

In a general geometric processing set up, the transformations T_g which have acted on an object are not known. Hence G-invariant descriptors are trivially preferable with respect to G-equivariant geometric descriptors. The latter in fact rely on the knowledge of the specific T_g applied. This highlights one of the advantages of the MVE coordinates over the LAP and NCC. In particular, for the investigated GTT task, the benefit of using MVE coordinates is illustrated in Fig. 5 where a chocolate tablet geometric texture $\mathcal{M}_S$ is applied to a face P of a cube object $\mathcal{M}_I$. The placing of $\mathcal{M}_S$ with respect to P requires a rotation in the first row and an isotropic scaling in the second row of Fig. 5. The use of LAP and NCC, being not invariant under rotation and scaling, introduces visible artifacts when the deformation contains rotation or scaling, while the use of MVE allows to preserve the orientation and the dimension of the structural details.

4.1. Inexact invariance under free-form deformation

The introduced invariance properties capture a geometry processing context where the texture transferring task only requires affine trasformations. In a more realistic GTT context a far larger set of transformations is involved. Inevitably, global, exact invariance needs to be replaced by a local, inexact one. In fact, we need to deal with 3D geometry textures that undergo non-rigid deformations when applied to free-form objects. Such deformations can be modelled as transformations that preserve as much as possible the intrinsic structure of the underlying (Riemannian) manifold, while transferring at best the descriptors captured by the geometric texture.

Let D denote the space of Riemannian manifolds, and $\mathcal{M}_S, \mathcal{M}_I \in D$. Then the deformation of $\mathcal{M}_S$ onto $\mathcal{M}_I$ is a family of smooth and invertible maps $\eta : D \to D$ that is modelled by the combination of an affine-transformation T_g that aligns the domains in a way that the corresponding structures are best preserved, and a non-rigid deformation T_τ to mold $\mathcal{M}_S$ on the structure of $\mathcal{M}_I$

$$\eta = T_\tau \circ T_g. \tag{20}$$

We can now extend the previous definitions of exact invariance and equivariance under group actions with a softer notion of deformation stability (or inexact invariance):

$$\|f(T_\eta x) - f(x)\| \leq C c(\tau)\|x\|, \quad \forall x \in D \tag{21}$$

where C is some constant independent of x, and the cost measure $c(\tau)$ is such that $c(\tau) = 0$ whenever $\tau \in G$, thus generalising the G-invariance property. A function f satisfying the above equation is said to be geometrically stable. Although $c(\tau)$ is difficult to measure, we will incorporate (21) in the variational formulation proposed for the GTT solution in Section 5. In the context of GTT, a practical way to define a deformation cost is to consider the discrepancy between the generated GTT textured surface $\mathcal{M}_T$, and a reference ground truth textured surface $\mathcal{M}_{GT}$. In general, a quantitative evaluation of the discrepancy between a manifold $\Omega \in D$ and another manifold $\bar{\Omega} \in D$ is the Hausdorff distance $d_H(\Omega, \bar{\Omega})$ between the two meshes, defined as

$$d_H(\Omega, \bar{\Omega}) = \max\left\{\sup_{x\in\Omega}\inf_{y\in\bar{\Omega}} d(x,y), \quad \sup_{y\in\bar{\Omega}}\inf_{x\in\Omega} d(y,x)\right\}, \tag{22}$$

with $d(x, y)$ being the Euclidean distance between vertices x and y.

Let $\mathcal{M}_T$ be generated by GTT exploiting the geometric descriptors f, named GTT-f, then GTT-f is stable if $d_H(\mathcal{M}_{GT}, \mathcal{M}_T)$ is small, being $d_H(\mathcal{M}_{GT}, \mathcal{M}_T) = 0$ when the two manifolds are isometric.

5. Variational models for geometric texture transfer

The GTT is obtained by replacing the coarse patch $P \subset \mathcal{M}_I$ with the finer mesh $\mathcal{M}_S$ whose n_S vertices must be appropriately placed in the new locations V^* by an unknown deformation η defined in (20), thus preserving at best the shape of $\mathcal{M}_I$ while adding the geometric textured details of $\mathcal{M}_S$. The resulting textured mesh $\mathcal{M}_T$ has vertices $V_T = (V_I \setminus P) \cup V^*$. We propose to determine the vertex positions V^* through a variational approach, by minimizing the sum of a reconstruction term $R(V)$, that aims at preserving the details of the texture, with a soft constraint term $g(V)$, that forces V^* to recover the global shape of the patch P. This reads as

$$V^* = \arg\min_V \{R(V) + \lambda_C g(V)\}, \tag{23}$$

where $\lambda_C := \lambda \cdot 1_C(V)$, $1_C(V)$ is the indicator function of a subset C of vertices of V_S with value 1 at points of C and 0 at points of $V_S \setminus C$, $\lambda \in \mathbb{R}$ is a positive parameter. The set V^* has the same number of vertices n_S as $\mathcal{M}_S$ and inherits its topological structure, and thus it represents the details with the appropriate fine mesh resolution. The term $R(V)$ depends on the particular geometric descriptor involved, then $R(V)$ can be formulated as the inverse reconstruction problems (5), (9) or (15), depending on whether the transfer is obtained via LAP, NCC or MVE descriptors, respectively. This choice leads to the following three alternative variational problems

$$V^* = \arg\min_{V\in\mathbb{R}^{n_S\times 3}} \{\, \mathcal{J}_1(V) := \frac{1}{2}\|L_w V - \delta_L\|_2^2 + \lambda_C g(V)\}, \tag{24}$$

$$V^* = \arg\min_{V\in\mathbb{R}^{n_S\times 3}} \{\, \mathcal{J}_2(V) := \frac{1}{2}\|N_w V - \delta_N\|_2^2 + \lambda_C g(V)\}, \tag{25}$$

$$V^* = \arg\min_{V\in\mathbb{R}^{n_S\times 3}} \{\, \mathcal{J}_3(V) := \frac{1}{2}\|V - F(V; w, b)\|_2^2 + \lambda_C g(V)\}, \tag{26}$$

where w, b in (26) refer to the source mesh $\mathcal{M}_S$. The three variational models (24),(25), and (26) share the same soft constraint term $g(V)$, and Dirichlet boundary conditions which not only allow to correctly align the boundaries b_P of the replaced patch P with b_S of $\mathcal{M}_S$, but also to uniquely determine the set of vertices V^*.

Details on boundary setting are given in Section 5.1, the discussion on the soft constraints will be given in Section 5.2, the overall GTT algorithm will be finally described in Section 6.4.

5.1. Boundary setting

The boundary setting involves both the boundary alignment between the boundary b_S of the geometric texture $\mathcal{M}_S$ and the boundary b_P of the patch P, and the boundary conditions for the operators in (23). The mild assumption formulated for a correct GTT is that b_P and b_S are polygonal approximants of counter clock-wise oriented, closed, simple curves in $\mathbb{R}^3$ with the same number of vertices. If this is not the case, a simple refinement preprocessing is applied by adding $n_{b_S} - n_{b_P}$ new vertices to b_P, uniformly distributed among its edges. Hence a bijection between b_P and b_S is easily established by imposing the correspondence of a couple of adjacent vertices between b_S and b_P, either by user interaction - in case the geometric texture must pursue a predetermined orientation - or automatically, via random assignment. This allows to define the starting point between the two polygonals, since they already follow a common orientation.

The common shared boundary is then imposed as Dirichlet boundary conditions to obtain full-rank matrices L_w in (5) and N_w in (9), and a unique solution for (15).

Finally, distortions due to a different proportional ratio between the geometric texture $\mathcal{M}_S$ and the target patch P can be avoided by uniformly scaling $\mathcal{M}_S$ of a factor $|b_P|/|b_S|$, with $|b_P|$ and $|b_S|$ the length of the boundaries.

5.2. Soft constraints

The reconstruction process in (24),(25), and (26) is driven by soft constraints on the internal vertices of $\mathcal{M}_S$ to recover at best the underlying shape of the original patch P. At this aim, a conformal parameterization is applied to both the patch $P \subset \mathcal{M}_I$ and the mesh $\mathcal{M}_S$ onto a common circular parametric domain Ω bounded by $b_S \equiv b_P$.

Given a rate $r \in (0, 100)$, eventually definable by a potential user, we aim at establishing a bijection between $m = \lfloor n_P \cdot r/100 \rfloor << n_V$ vertices of P, randomly chosen, and m vertices of $\mathcal{M}_S$. Let $C = \{(k_i, j_i) : k_i$ is vertex index of P, j_i is vertex index of $\mathcal{M}_S, i = 1, \ldots, m\}$ be the subset of indices of vertices of P and associated vertices of $\mathcal{M}_S$ closest to them in the parametric domain Ω. Each $v_{j_i} \in \mathcal{M}_S$ is associated with a vertical displacement ϵ_{j_i}. Then, for each selected vertex $p_{k_i} \in P$, with associated normal $\tilde{n}_{k_i}$, the position of $\bar{v}_{j_i}$ is given by

$$\bar{v}_{j_i} = p_{k_i} + \epsilon_{j_i}\tilde{n}_{k_i}. \tag{27}$$

The set of the new handles $\bar{V} = \{\bar{v}_{j_i}\}_{i=1}^m$ defines the soft constraint term $g(V)$ which aims to preserve the shape of P while adding the texture displacement.

The role of the indicator function λ_C in (24), (25), and (26) is played upon discretization by a mask matrix $\Lambda \in \{0, \lambda\}^{m\times n_S}$ of non-zeros only in correspondence to constraints C, explicitly defined as

$$\Lambda_{ij} = \begin{cases} \lambda & \text{if } j = j_i \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

The displacement energy term thus reads as follows

$$\lambda_C g(V) = \sum_{i=1}^{m} \lambda (v_{j_i} - \bar{v}_{j_i})^2 = \|\Lambda(V - \bar{V})\|_2^2, \tag{29}$$

which forces the vertices $v_{j_i} \in \mathcal{M}_S$ towards the positions of the corresponding vertices p_{k_i} of the patch P, with a displacement ϵ_{j_i} in its normal direction $\tilde{n}_{k_i}$.

Whenever the geometric texture $\mathcal{M}_S$ is a 3D mesh that cannot be expressed as a height field, to compute the handle vertices in $\bar{V}$ in the soft constraints (29), formula (27) is replaced by the following

$$\bar{v}_{j_i} = p_{k_i} + d_x^{j_i} t_{k_i} + d_y^{j_i} b_{k_i} + d_z^{j_i}\tilde{n}_{k_i}, \tag{30}$$

where $d^{j_i} = (d_x^{j_i}, d_y^{j_i}, d_z^{j_i})$ is the local displacement vector on $\mathcal{M}_S$, and $(t_{k_i}, b_{k_i}, \tilde{n}_{k_i})$ is the local frame at p_{k_i} on the patch P.

We refer the reader to the Example 7.1 which validates the shape preserving reconstruction capability of the variational models (25) and (26) when applied to the solution of the GTT task with varying r factors. For the overall examples considered a very low r value proved to be sufficient for a good quality GTT.

6. Numerical optimization of the variational GTT models

In this section we illustrate the details of the efficient optimization algorithms proposed to numerically solve the Laplacian Model (24), the NCC Model (25), and the MVE Model (26). Finally in Section 6.4 we outline the main steps of the overall algorithm GTT.

6.1. Solution of the Laplacian model (24)

The optimization problem (24) can be rewritten as the following linear least squares problem

$$V^* \in \arg\min_V \mathcal{J}_1(V) := \frac{1}{2}\|AV - B\|_2^2 \tag{31}$$

with

$$A := \begin{bmatrix} L_w \\ \Lambda \end{bmatrix} \in \mathbb{R}^{(n_S+m)\times n_S}, \quad B := \begin{bmatrix} \delta_L \\ \Lambda \bar{V} \end{bmatrix} \in \mathbb{R}^{(n_S+m)\times 3}, \tag{32}$$

$L_w \in \mathbb{R}^{n_S \times n_S}$ defined in (4), and Λ defined in (28). In matrix B, $\delta_L \in \mathbb{R}^{n_S\times 3}$ contains the Laplacian coordinates of the geometric texture $\mathcal{M}_S$ and $\bar{V} \in \mathbb{R}^{m\times 3}$ represents the constraints set defined in (27) or (30). Under the Dirichlet boundary conditions on b_S, matrix A is full column rank, A^TA is non-singular, and thus the unique minimizer of (24) can be computed by solving the standard normal equations $A^TAV = A^TB$.

The computational cost for the solution of (31)-(32) corresponds to the solution of three linear systems for each coordinate vector $V = (V_x, V_y, V_z)$, with the same coefficient matrix and right hand-side defined by the corresponding three columns of B.

6.2. Solution of the NCC model (25)

The numerical solution of the NCC-based variational problem (25) is obtained analogously to the numerical solution of (24) by rewriting the linear least squares problem as

$$V^* \in \arg\min_V \mathcal{J}_2(V) := \frac{1}{2}\|AV - B\|_2^2, \qquad A = \begin{bmatrix} N_w \\ \Lambda \end{bmatrix}, \quad B = \begin{bmatrix} \delta_N \\ \Lambda \bar{V} \end{bmatrix} \tag{33}$$

with $A \in \mathbb{R}^{(n_S+m)\times n_S}$ and $B \in \mathbb{R}^{(n_S+m)\times 3}$, where δ_N are the NCC coordinates of the geometric texture $\mathcal{M}_S$ and N_w is the sparse NCC matrix defined in (8). Under Dirichlet boundary conditions on b_S, matrix A is full column rank, then the unique minimizer is obtained by solving the system of normal equations $A^TAV = A^TB$.

Although the use of Laplacian (differential) LAP or NCC descriptors forces local detail preserving, the shape of such details appears, in general, deformed since these descriptors are not invariant with respect to rotation and scaling. At this aim, in [27] the authors proposed a rotated Laplacian reconstruction approach which relies on the estimate of the local frame rotations matrix $\mathcal{R}$ in order to exploit the equivariance property proved in Prop. 4.1

$$N_w(\mathcal{R}(V)) = \mathcal{R}(N_w(V)).$$

A simpler but effective idea was proposed in [4] where an iterative approach alternates the solution of the normal equations (33) with an update of the NCC coordinates (and, therefore, of the matrix B). In particular, starting from $\delta^{(0)} = \delta_N$, by recalling the coordinates NCC are parallel to the normals to the vertices, $V^{(k)}$ and $\delta^{(k)}$ are updated through the following iterative scheme:

$$\begin{cases} \text{Solve } A^TAV = A^TB^{(k)} \text{ for } V^{(k+1)}; \\ \delta_i^{(k+1)} = sgn((\delta_N)_i)\,\|(\delta_N)_i\|\,n_i, \text{ with } n_i \text{ normal to the vertex } v_i^{(k+1)}, \quad \forall i. \end{cases} \tag{34}$$

This correction to the naive reconstruction method, makes the final result less dependent on the original orientation of the source mesh $\mathcal{M}_S$.

The computational cost of one iteration of the NCC updating scheme (34) consists of the solution of three linear systems, the estimate of the vertex normal and the update of the directions $\delta^{(k+1)}$; all of which depend on the number of vertices n_S.

6.3. Solution of the MVE model (26)

The vertex set V^* of the target mesh $\mathcal{M}_T$, corresponding to the region of interest $P \in \mathcal{M}_I$, is obtained as solution of a nonlinear least squares (NLLS) problem. In particular, let $n = n_S - |b_S|$, be the cardinality of the sought vertex set V^*, we define the differentiable nonlinear residual vector function $r : \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ as $r(V) = V - F(V)$, with components $r_i(V) \in \mathbb{R}^3$, $r_i(V) = (r_i^{(x)}(V), r_i^{(y)}(V), r_i^{(z)}(V))^T$ the nonlinear residual at vertex v_i, defined as

$$r_i(V) := v_i - F_i(V), \quad i = 1, \ldots, n. \tag{35}$$

We want to solve the NLLS minimization problem

$$V^* = \arg\min_{V\in\mathbb{R}^{3n}} \mathcal{J}_3(V) := \frac{1}{2}\|r(V)\|_2^2 + \lambda_C\|V-\bar{V}\|_2^2. \tag{36}$$

For the purpose of developing a numerical solution of the NNLS problem, we define the Jacobian of the residual vector function $r(V)$ as the 3×3 block matrix

$$J(V) = \begin{bmatrix} \frac{\partial r^{(x)}(V)}{\partial V^{(x)}} & \frac{\partial r^{(x)}(V)}{\partial V^{(y)}} & \frac{\partial r^{(x)}(V)}{\partial V^{(z)}} \\ \frac{\partial r^{(y)}(V)}{\partial V^{(x)}} & \frac{\partial r^{(y)}(V)}{\partial V^{(y)}} & \frac{\partial r^{(y)}(V)}{\partial V^{(z)}} \\ \frac{\partial r^{(z)}(V)}{\partial V^{(x)}} & \frac{\partial r^{(z)}(V)}{\partial V^{(y)}} & \frac{\partial r^{(z)}(V)}{\partial V^{(z)}} \end{bmatrix} \in \mathbb{R}^{3n\times 3n} \tag{37}$$

with blocks $[J(V)]^{kl} \in \mathbb{R}^{n\times n}$, $k,l = 1,2,3$, and we summarize the main properties of $J(V)$.

Proposition 6.1. *The square matrix $J(V)\in\mathbb{R}^{3n\times 3n}$ in (37) is full rank, positive definite and it is highly sparse having at each row corresponding to i-th vertex at most $3|\mathcal{N}(v_i)|+1$ non-zero elements. The elements (i,j), $i,j=1,\dots,n$ for all blocks $[J(V)]^{kl}$, $k,l\in\{1,2,3\}$, are defined as*

$$\begin{aligned} [J(V)]^{kl}_{i,i} &= \left[\tfrac{\partial r_i(V)}{\partial v_i}\right]^{kl} = I_3, \\ [J(V)]^{kl}_{i,j} &= \left[\tfrac{\partial r_i(V)}{\partial v_j}\right]^{kl} = -w_{ij}I_3 + \frac{w_{ij}b_{ij}}{\|z-Q_1v_j\|_2}\, n_i\left(Q_1^T(z-Q_1v_j)\right)^T + \\ &\quad - \sum_{k\neq j}\frac{w_{ik}b_{ik}}{\|Q_2v_j+z-N_iv_k\|_2}\, n_i\left(Q_2^T(Q_2v_j+z-N_iv_k)\right)^T, \end{aligned} \tag{38}$$

where

$$Q_1 = (1-w_{ij})N_i, \qquad Q_2 = w_{ij}N_i, \qquad z = N_i\sum_{l\neq j} w_{il}v_l\ .$$

Proof. To derive an explicit expression for $J(V)$, we first observe that $\frac{\partial r_i(V)}{\partial v_j}\in\mathbb{R}^{3\times 3}$, and $\frac{\partial r_i(V)}{\partial v_j} = 0$ for $j\notin\mathcal{N}(v_i)$. Next, we can rewrite $r_i(V)$ by separating the neighbor v_j from the other neighbors $v_k\in\mathcal{N}(v_i)$, as

$$r_i(V) = v_i - w_{ij}\big(v_j + \|z-Q_1v_j\|_2 b_{ij}n_i\big) - \sum_{k\neq j} w_{ik}\big(v_k + \|Q_2v_j+z-N_iv_k\|_2 b_{ik}n_i\big). \tag{39}$$

Then, by applying in (39) the chain rule $\frac{\partial}{\partial x}\|Ax-b\|_2 = \frac{A^T(Ax-b)}{\|Ax-b\|_2}$ for $x=v_j$, we get (38). □

An approximate solution for the minimization problem (36) can be obtained by imposing the first-order optimality conditions:

$$J^T(V)r(V) + 2\lambda_C(V-\bar{V}) = 0, \tag{40}$$

and then using a traditional optimization method, such as the Newton-Raphson method to solve (40). Alternatively, a well-assessed numerical approach for addressing directly the minimization problem (36) is the Gauss-Newton method, which does not require the computation of second order derivatives. Essentially, it is based on the approximation of the Hessian matrix of $\mathcal{J}_3(V)$ in (36) with $\nabla^2\mathcal{J}_3(V)\approx J(V)^TJ(V)+2\lambda_C$, by ignoring all the second order terms from $\nabla^2\mathcal{J}_3(V)$. Gauss-Newton method iterates from an initial guess $V^{(0)}$ and performs a line search along the direction $p_{GN}^{(k)}$ determined by solving the following linear system

$$(J^T(V^{(k)})J(V^{(k)}) + 2\lambda_C)\,p_{GN} = -J^T(V^{(k)})r(V^{(k)}) - 2\lambda_C(V^{(k)}-\bar{V}). \tag{GN}$$

The coefficient matrix is symmetric positive definite and has dimension $3n\times 3n$. The new vertex positions is then updated as

$$V^{(k+1)} = V^{(k)} + \alpha^{(k)}p_{GN}, \tag{41}$$

with an adaptive step-size $\alpha^{(k)}$ obtained via line search with backtracking, satisfying Armijo condition:

$$\mathcal{J}_3\big(V^{(k)}+\alpha p_{GN}\big) - \mathcal{J}_3\big(V^{(k)}\big) \le -2\beta\,\alpha\,(p_{GN})^T\nabla\mathcal{J}_3\big(V^{(k)}\big) \tag{42}$$

with $\beta\in[0,1)$ fixed parameter.

Due to the nonlinear nature of this model, an initial guess $V^{(0)}$ sufficiently close to the sought solution would allow for a fast convergence in a few iterations. Nevertheless, in our experiments, we noticed that the scheme is robust and converges towards the expected solution even for $V^{(0)}$ being located far away, e.g. $V^{(0)}$ being $\mathcal{M}_S$ in its original position. Favourable

choices for $V^{(0)}$ can be either the minimal surface fit for given boundary b_P or the least squares solution of the GTT-NCC model (25), due to the similarity between weights in MVE and in NCC.

Assuming the contribution of derivatives by non-matching coordinates is negligible, i.e. $[J(V)]^{kl} \approx 0$, if $k \neq l$, $k, l = 1, 2, 3$, we can further simplify $J(V)$ into a block-diagonal matrix by neglecting the non-diagonal blocks, thus making the Jacobian matrix separable for each spatial coordinate. This gives raise to the solution of three highly sparse linear systems of the form (41). The experiments support the above assumption.

6.4. Algorithm GTT

We summarize in Algorithm 1 the main steps of the GTT procedure for the three different variational models proposed, which will be named GTT-LAP, GTT-NCC and GTT-MVE according to the respective choice of descriptors.

Algorithm 1 Geometric Texture Transferring.

Input: · base mesh $\mathcal{M}_I = (V_I, E_I, T_I)$, with patch $P \subset \mathcal{M}_I$
· geometric texture mesh $\mathcal{M}_S = (V_S, E_S, T_S)$
Output: · textured mesh $\mathcal{M}_T = (V_T, E_T, T_T)$ with $V_T = (V_I \setminus P) \cup V^*$
Parameters: · $r \in (0, 100)$, rate of vertices chosen as soft constraints
· $\lambda > 0$ soft-constraint parameter
· corresponding vertices in b_S and b_P, in case of manual alignment

Preliminary set up:

- **Dirichlet boundary conditions**: refinement of b_P (if needed), bijection between b_S and b_P (manually or automatically);
- **soft constraint term** $g(V)$: parameterization of$P$ and $\mathcal{M}_S$, random choice of $\lfloor n_P \cdot r/100 \rfloor$ vertices of P, correspondence with vertices of$\mathcal{M}_S$, definition of $\tilde{V}$ through (30) or (27);
- **reconstruction term** $R(V)$: computation of thedescriptors LAP, NCC or MVE from the source mesh $\mathcal{M}_S$, for thesetting of each $R(V)$ and the three models: (24),(25),(26).

Variational Models Solution:

GTT-LAP (24)	Construction of matrices A and B in (32) solve for V the normal equations $A^TAV = A^TB$
GTT-NCC (25)	Construction of matrices A and B in (33) *Until stopping criterion in (43) is satisfied* · solve for V the normal equations $A^TAV = A^TB$ · update δ_N and B as in (34)
GTT-MVE (26)	Set an initial guess $V^{(0)} = V_S$ *Until stopping criterion in (43) is satisfied* · compute p_{GN} from (GN) · compute step-size $\alpha^{(k)}$ via line-search asin (42) (or use a constant α) · update V as in (41)

For height-map geometric texture, the mesh $\mathcal{M}_S$ is generated from a 2D image by setting the vertices in V_S to be pixel-centered.

The influence of parameters r and λ on the quality of the GTT results is discussed in Section 7.1 and Section 7.2, respectively, which suggests satisfactory results can be achieved by selecting $\lambda = 10$ for GTT-LAP and GTT-NCC and $\lambda = 1$ for GTT-MVE, while setting the rate $r = 20$.

We terminate the iterations of the GTT algorithm, in case iterative procedures are used for GTT-NCC and GTT-MVE, as soon as either of the two following stopping criteria is satisfied

$$\frac{\left\|V^{(k)} - V^{(k-1)}\right\|_2}{\left\|V^{(k)}\right\|_2} < 10^{-4}, \qquad \frac{\left\|\mathcal{J}(V^{(k)}) - \mathcal{J}(V^{(k-1)})\right\|_2}{\left\|\mathcal{J}(V^{(k)})\right\|_2} < 10^{-4}. \tag{43}$$

7. Numerical GTT examples

In this section we illustrate GTT examples to validate the overall procedure sketched in Algorithm 1 and compare among the performance of the GTT-LAP, GTT-NCC and GTT-MVE variational models. The algorithms has been implemented in MatLab R2021a and executed on a laptop with 2.10 GHz AMD Ryzen 5 quad-core processor and 16 GB 2.4 MHz RAM.

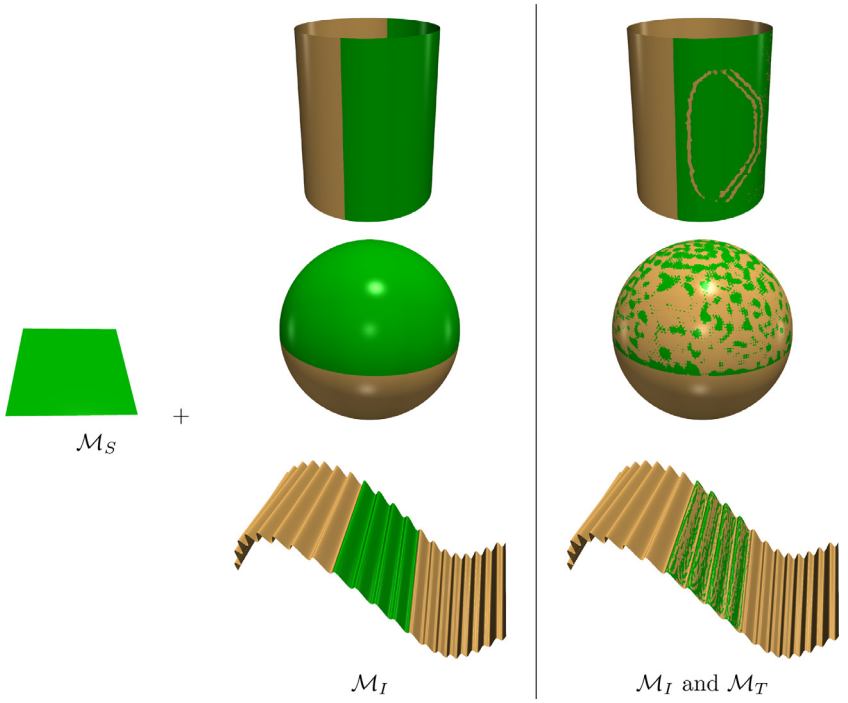


Fig. 6. Example 1: shape preservation in GTT computation, for a flat geometric texture $\mathcal{M}_S$ transferred to $P \subset \mathcal{M}_I$ in green (middle column). The resulting $\mathcal{M}_T$ overlapped with base mesh $\mathcal{M}_I$ (right column). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Example 1: Hausdorff distances to ground truth patch P using flat geometric texture $\mathcal{M}_S$ defined on a uniform grid of dimensions 150×150 vertices.

GTT-NCC		$d_H(\mathcal{M}_I, \mathcal{M}_T)$	
r%	cylinder+flat	sphere+flat	free-form+flat
1	5.83×10^{-3}	1.59×10^{-2}	1.03×10^{-1}
5	1.42×10^{-3}	4.67×10^{-3}	4.36×10^{-2}
20	6.45×10^{-4}	2.09×10^{-3}	2.14×10^{-2}
50	3.87×10^{-4}	1.25×10^{-3}	8.25×10^{-3}
80	3.23×10^{-4}	1.06×10^{-3}	5.40×10^{-3}
95	2.93×10^{-4}	1.01×10^{-3}	4.59×10^{-3}
99	2.93×10^{-4}	1.01×10^{-3}	4.47×10^{-3}
GTT-MVE		$d_H(\mathcal{M}_I, \mathcal{M}_T)$	
r%	cylinder+flat	sphere+flat	free-form+flat
1	5.89×10^{-3}	1.59×10^{-2}	9.73×10^{-2}
5	1.49×10^{-3}	4.67×10^{-3}	2.46×10^{-2}
20	6.27×10^{-4}	2.09×10^{-3}	1.55×10^{-2}
50	2.81×10^{-4}	1.25×10^{-3}	3.28×10^{-3}
80	2.43×10^{-4}	1.07×10^{-3}	2.64×10^{-3}
95	2.05×10^{-4}	1.07×10^{-3}	2.02×10^{-3}
99	1.99×10^{-4}	1.07×10^{-3}	2.02×10^{-3}

7.1. Example 1: Shape preserving GTT

To validate the shape preserving capability of the proposed variational models, we investigate qualitatively and quantitatively the effect of transferring a flat geometric texture $\mathcal{M}_S$. The expected result is a mesh $\mathcal{M}_T$ with the same exact shape as the base mesh $\mathcal{M}_I$, but with a different resolution in the patch region P, inherited from the geometric texture $\mathcal{M}_S$. From a qualitative point of view in Fig. 6 we show the results for three different meshes $\mathcal{M}_I$, and a set C of soft constraints $g(V)$ obtained with a rate $r = 20\%$ of vertices.

In Table 1 we report the Hausdorff distances $d_H(\mathcal{M}_I, \mathcal{M}_T)$, defined in (22), for different values of the rate r of randomly chosen vertices in C. Although for sphere and cylinder shapes the results from GTT-NCC and GTT-MVE are quite similar, the measured distances for GTT-NCC double in case of free-form surface $\mathcal{M}_I$ as the one in Fig. 6, third row. As expected, in general, when the percentage r increases also the corresponding accuracy increases, or better, the Hausdorff distance

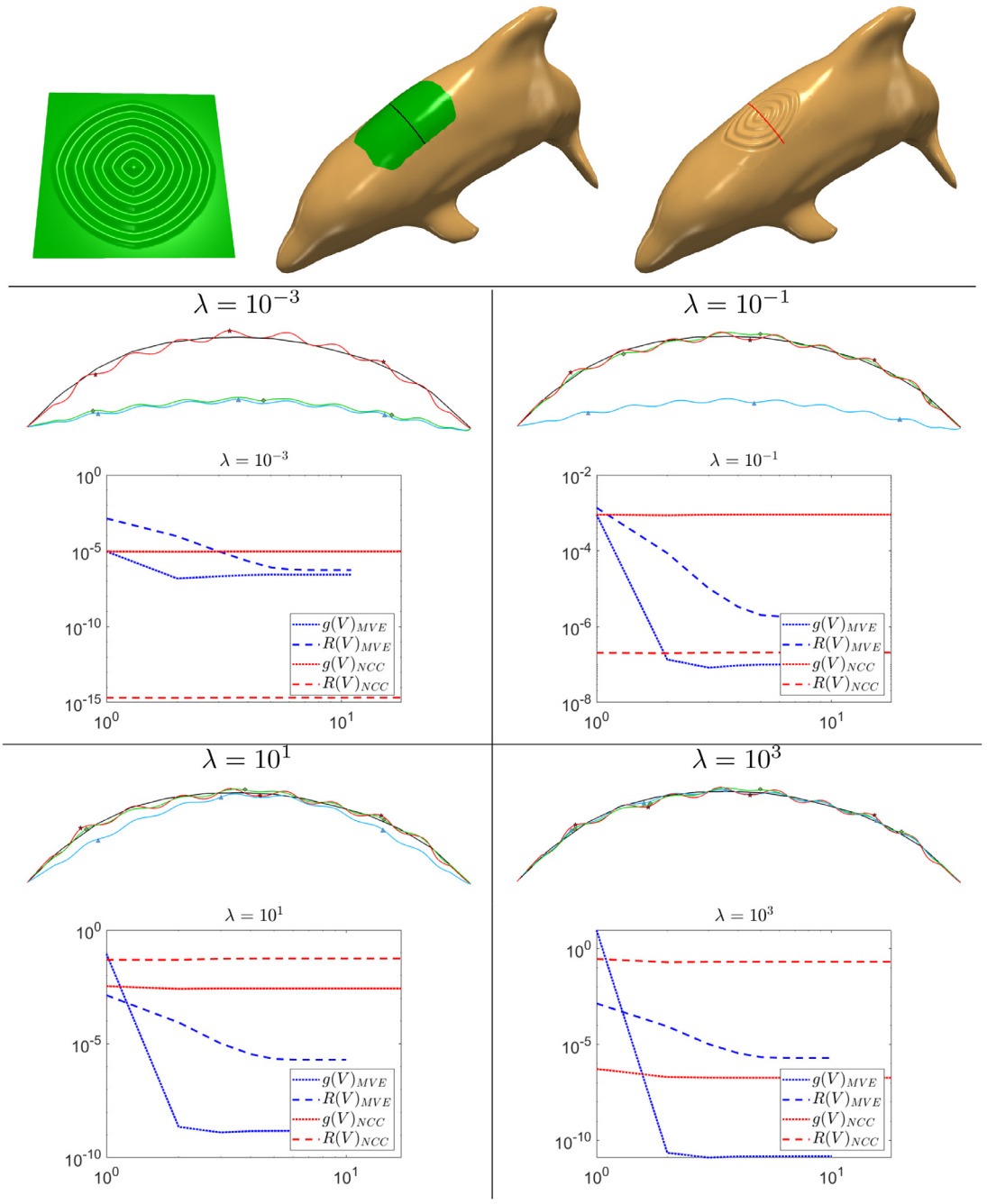


Fig. 7. Example 2: influence of the λ parameter. First row: a GTT example with $\mathcal{M}_S$, $\mathcal{M}_I$ and $\mathcal{M}_T$. Second and third rows: cross-sections of the P patch (in black), of GTT-LAP (in green, marked $\diamond$), of GTT-NCC (in blue, marked $\triangle$), and of GTT-MVE (in red, marked $\star$) and plots of the reconstruction term $R(V)$ and the constraint term $\lambda_C g(V)$ (related to GTT-NCC and GTT-MVE), for increasing values of λ. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

decreases so that the accuracy of the models is reasonably high for high rates of r. However, the shape of the patch P is well preserved even for low rates r of vertices in the soft constraints term $g(V)$.

7.2. Example 2: How the λ parameter affects the shape preserving

The second example analyses how the value of the soft constraint parameter λ in (23) affects the final GTT results obtained from the three variational models. The rate r is fixed to 20% in accordance with the results shown in Example 1.

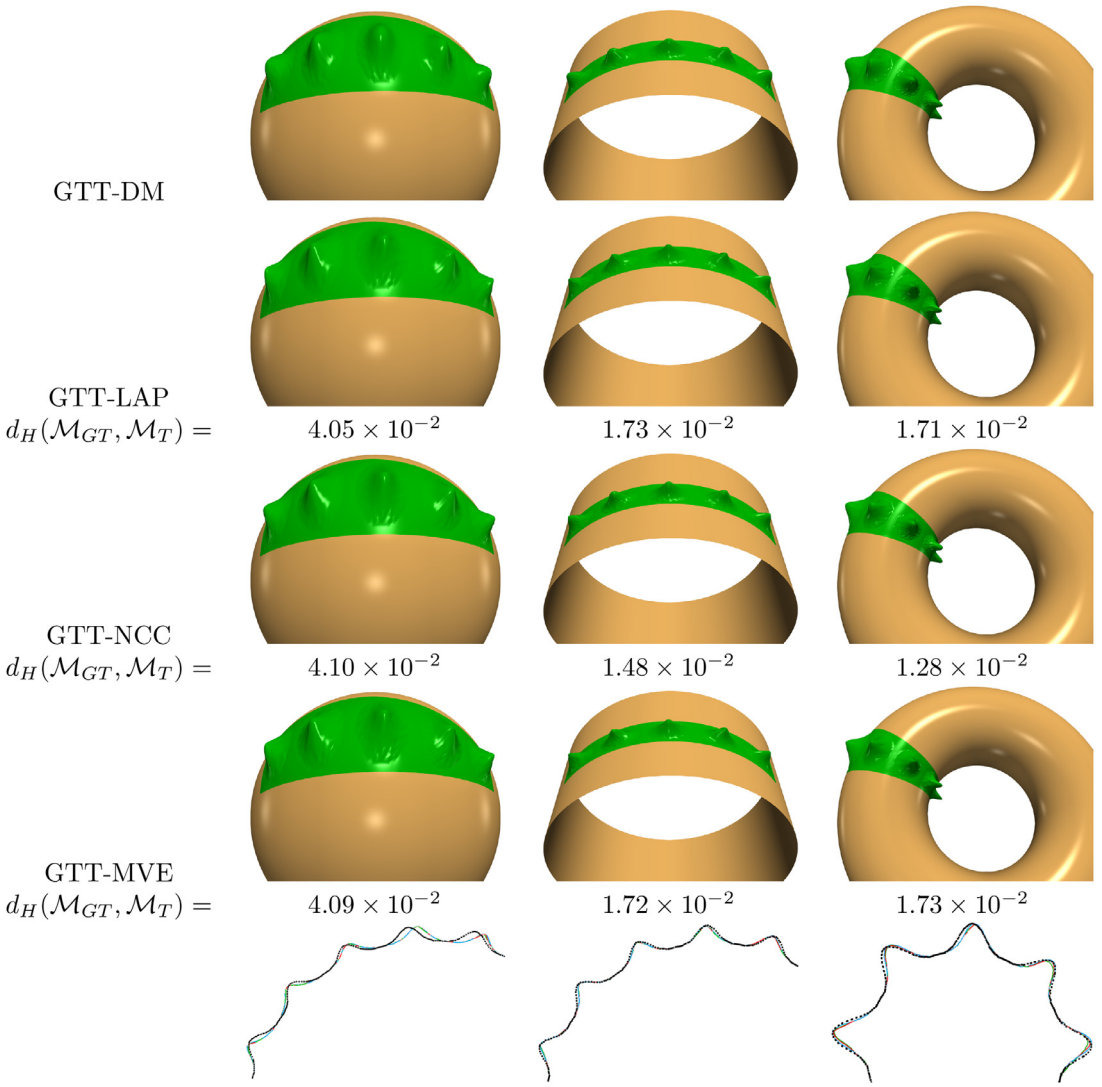


Fig. 8. Example 3: Comparison with displacement mapping (DM) on parametric surfaces (sphere, cylinder, torus). In the last row, corresponding slices: DM in dotted black, LAP in green, NCC in blue, MVE in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

At this aim an example of a geometric texture $\mathcal{M}_S$ transferred to a dolphin mesh $\mathcal{M}_I$ is illustrated in Fig. 7, first row. In the second and fourth rows (top of panels), the cross-section curves from GTT-LAP (in green, with $\diamond$), GTT-NCC (in blue, $\triangle$) and GTT-MVE (in red, $\star$) results are shown for increasing values of λ. Finally, in Fig. 7, third and fifth row (bottom of panels), we report the corresponding plots of the terms $R(V)$ and $\lambda_C g(V)$ in (23), for the two iterative algorithms GTT-NCC and GTT-MVE in red and blue color, respectively.

For small λ values both GTT-LAP (24) and GTT-NCC (25) models perform poorly in reproducing the shape of the underlying patch P. This behaviour, in particular for GTT-NCC, is confirmed by the blue colored plots of its related terms $R(V)$ and $\lambda_C g(V)$: $R(V)$ value indicates perfect reconstruction, nevertheless the shape-preserving term $\lambda_C g(V)$ remains at high values. As λ value increases, the results of GTT-LAP and GTT-NCC improve, as illustrated by the green ($\diamond$) and blue ($\triangle$) cross-sections, respectively, in Fig. 7. However, only for a high λ value, $\lambda = 10^3$, the GTT-NCC model (25) is forced to follow the shape of P as $g(V)$ attains lower value with respect to $R(V)$. Nevertheless, under a close visual inspection of the cross-sections, we notice that the reconstruction does not reproduce the original oscillations. This is due to the high penalization imposed to satisfy the soft constraints which restraints the reconstruction term. Better performance is instead obtained in case of GTT-MVE reconstruction.

For what concerns the GTT-MVE (26) model, it proved to be extremely robust to variations of λ value, providing good reconstructions even for low λ values. This is qualitatively shown by the red cross-section in Fig. 7, and quantitatively by the convergence plots in the third row of Fig. 7 where the term $\lambda_C g(V)$ attains always lower values with respect to the reconstruction term $R(V)$.

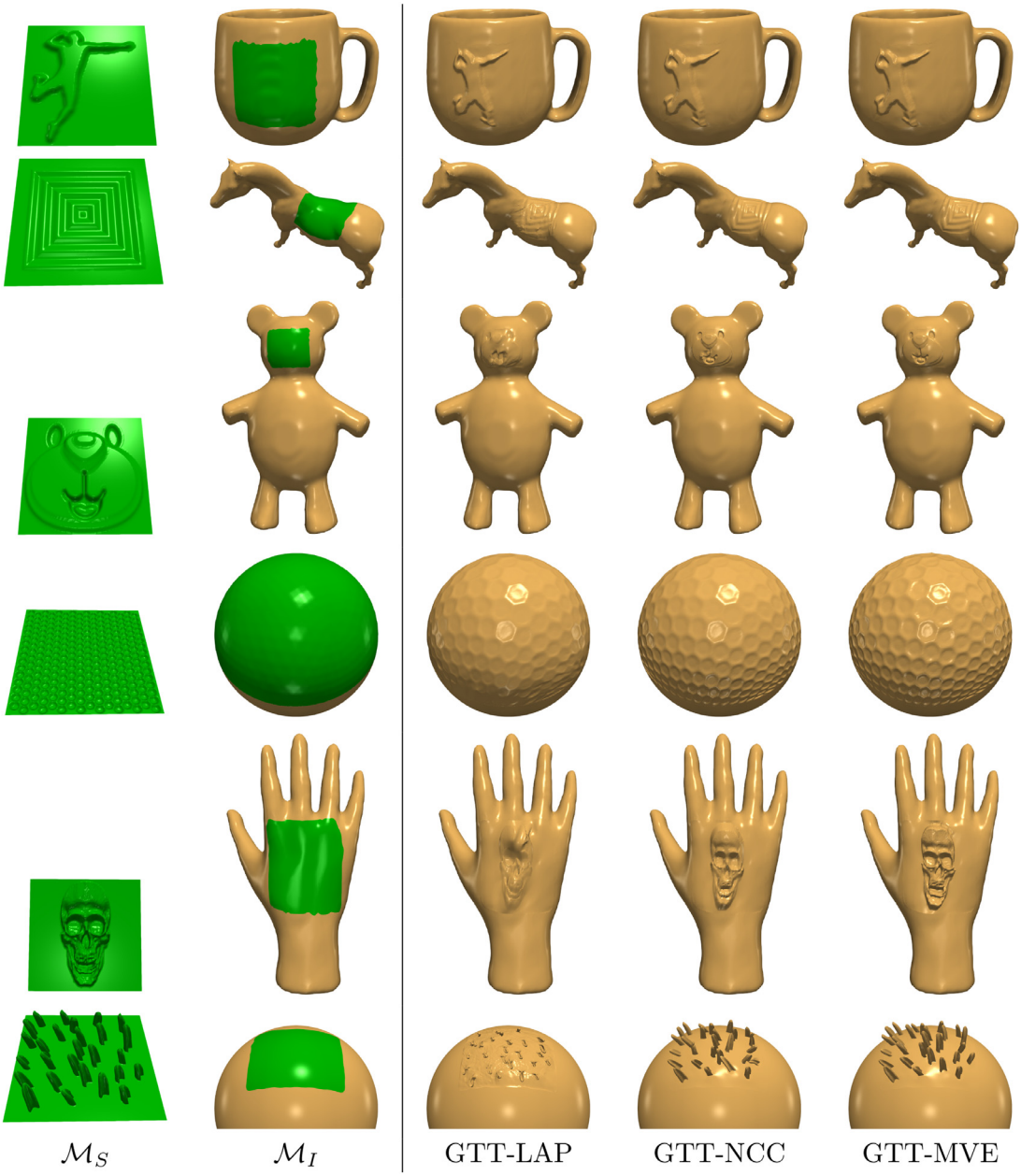


Fig. 9. Example 4: comparison among GTT-LAP (24), GTT-NCC (25) and GTT-MVE (26) algorithms applied to different meshes and different geometric textures.

7.3. Example 3: Comparison with parametric displacement mapping

It is quite easy to qualitatively assess whether the result of a GTT meets our expectations.

However, it is not entirely obvious what should be an ideal result on a free-form base surface. This may depend on subjective evaluations or on the application context. For the quantitative evaluation of the GTT results obtained by the three variational models, we considered three parametric surfaces as base meshes, in order to be able to perform an exact GTT, in terms of the standard displacement mapping formula (1), named GTT-DM in the following. This allowed us to evaluate a measure of discrepancy between the exact result $\mathcal{M}_{DT}$ and $\mathcal{M}_T$ by estimating the Hausdorff distance $d_H(\mathcal{M}_{DT}, \mathcal{M}_T)$.

Fig. 8 illustrates the results obtained via displacement mapping (GTT-DM) and with the three models GTT-LAP (24), GTT-NCC (25), GTT-MVE (26) for the base surfaces $\mathcal{M}_I$ sphere, cylinder and torus. We notice that the displacement mapping approach requires a bijective correspondence for all the parametric values of the height map that represents the geometric

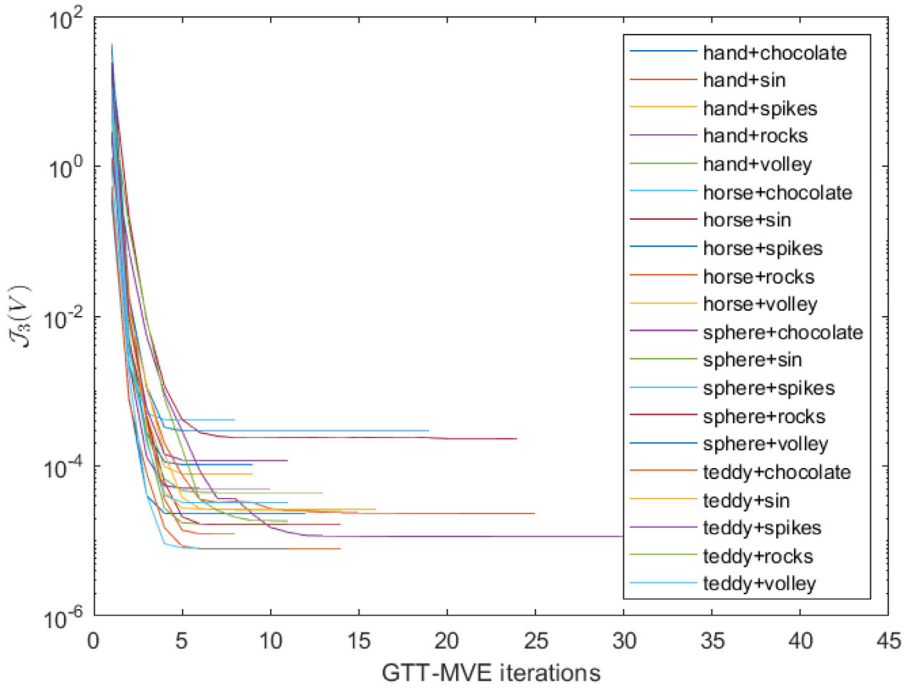


Fig. 10. Numerical convergence of $\mathcal{J}_3(V)$, related to the MVE model (26), when minimized by the Gauss-Newton algorithm for the solution of some examples in Fig. 11.

texture. In contrast, the proposed GTT models return the mesostructure $\mathcal{M}_T$ computing only a relatively small number of corresponding points.

Resulting Hausdorff values $d_H(\mathcal{M}_{DT}, \mathcal{M}_T)$, reported below each result in Fig. 8, confirm an overall good accuracy of the GTT-models GTT-LAP (24), GTT-NCC (25), GTT-MVE (26). However, in general, an exact GTT-DM cannot be applied to non-parametric surfaces, thus making it impossible to evaluate the GTT results quantitatively for arbitrary meshes.

7.4. Example 4: GTT performance

We finally evaluate GTT performance when applied to objects and geometric textures of different shapes to show the adaptability of the GTT algorithm to different geometric features and details.

We first show in Fig. 9 the results of the comparison between GTT-LAP, GTT-NCC and GTT-MVE which solve the variational problems in (24), (25) and (26), respectively. It is quite evident as the GTT-LAP - see third column in Fig. 9 - produces the worst quality geometric transfer results for any kind of surface and geometric texture. The non-linear GTT-MVE optimization model (26) provides the best local detail-preservation and it avoids possible self-intersections, see the fifth column of Fig. 9. On the other hand, the GTT-NCC gives overall good results, but it is more prone to producing artifacts, see the teddy bear face and skull textures, in the third and the fourth row of Fig. 9, respectively. In the sixth row of Fig. 9, the GTT results have been produced using a 3D geometric texture $\mathcal{M}_S$, not representable by a height map. In this case the soft constraints $g(V)$ have been defined following (30). Even in this case the GTT-MVE results preserve well the individual texture shape, while GTT-NCC fails to recover few of the "petal" shapes located close as well as far-left from the reader's point of view.

Additional results are reported in Fig. 11 which illustrates the behavior of the GTT-MVE algorithm applied to various patches $P \subset \mathcal{M}_I$, for various geometric textures $\mathcal{M}_S$. Each GTT result is accompanied by the average execution time in seconds and number of iterations of the Gauss-Newton method to emphasize its fast convergence for good quality results. We observe column-wise in Fig. 11 that the efficiency depends strongly both on the $\mathcal{M}_S$ resolution, reported below each geometric texture, and on the level of detail contained in $\mathcal{M}_S$. The shape of the patch P, illustrated row-wise in Fig. 11, has much less influence on the execution time. Certainly there is room for improvement in efficiency by adopting code optimization strategies, rather than the naive MATLAB implementation used.

We further investigated the empirical convergence of the Gauss-Newton iterative method for GTT-MVE model (26). To that aim, we run the optimization algorithm with initial guess $V^{(0)}$ set to be the original position of the texture mesh $\mathcal{M}_S$ and we stopped the algorithm as soon as one of the stopping criterion in (43) is satisfied under the tolerance 10^{-8}. We can observe the fast decreasing of the energy function $\mathcal{J}_3(V)$ in Fig. 10 for every GTT examples shown in Fig. 11.

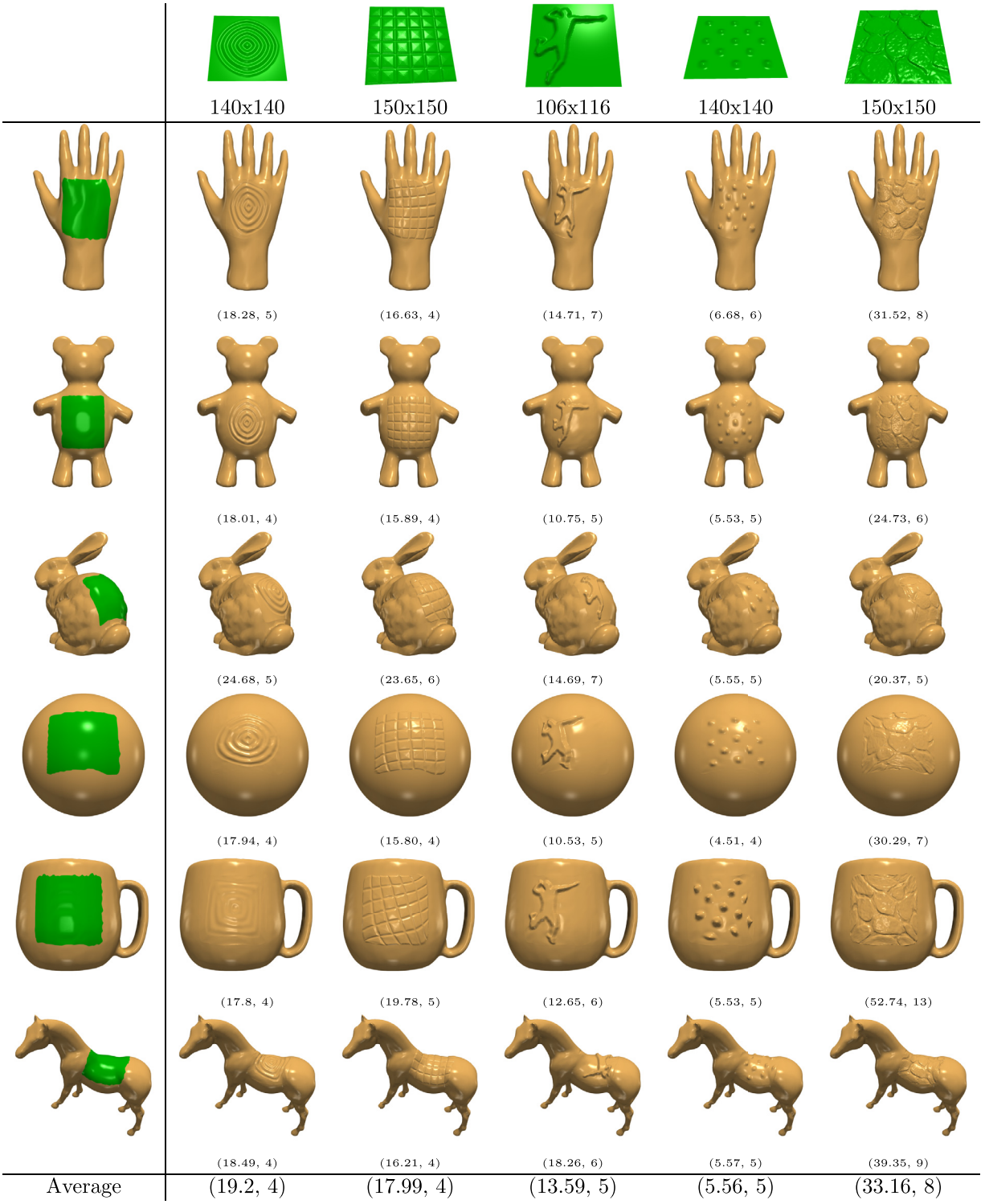


Fig. 11. Example 4: GTT-MVE results for meshes $\mathcal{M}_I$ in rows, geometric textures $\mathcal{M}_S$ in columns. In parenthesis, average execution time in seconds and number of iterations for each texture.

8. Conclusion

Geometric Texture Transfer, aimed to add finer details to surfaces, can be seen as a "vertex-texture mapping" where the values of the geometric texture source do not indicate alterations to pixel colors (as is common in computer graphics), but instead guide the change in vertex positions. Thus, enhancing this graphic task to be a particularly useful geometric modelling technique. Aim of this work has been to provide a numerical insight on the GTT problem by *first* investigating properties of local shape descriptors appropriated for this task to better capture the shape morphology, and, *second*, by formulating the GTT task as constrained variational linear/nonlinear optimization problems based on these descriptors. We analysed suitable numerical algorithms and several critical aspects in the solution of these minimization problems, such as the influence of the parameter λ, the rate of shape interpolant constraints r, and the influence of the invariance/equivariance under affine and non-affine deformations.

The GTT variational approaches proposed provide consistent values across boundaries between P and $\mathcal{M}_T$ under the mild assumption of the same number of vertices in the common boundary. In case boundary subdivisions are required to satisfy this condition, this may produce an unpleasant wrinkle along the boundary. A smoother joint can be obtained by considering suitable boundary overlap strategies, and boundary remeshing, which can be investigated in future work. Future research directions will be the embedding of the proposed GTT framework into deep learning techniques for texture synthesis, which generate 3D geometric textures.

Data availability

No data was used for the research described in the article.

Acknowledgement

Research was supported in part by the National Group for Scientific Computation (GNCS-INDAM), Research Projects 2022.

References

[1] J. Blinn, Simulation of wrinkled surfaces, in: Siggraph 1978, Association for Computing Machinery, Inc., 1978, pp. 286–292.
[2] L. Calatroni, M. Huska, S. Morigi, G.A. Recupero, A unified surface geometric framework for feature-aware denoising, hole filling and context-aware completion, J. Math. Imaging Vis. 65 (1) (2022) pp. 82–98.
[3] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, H.P. Seidel, Laplacian surface editing, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, in: SGP '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 175184.
[4] S. Wang, Y. Cai, Z. Yu, J. Cao, Z. Su, Normal-controlled coordinates based feature-preserving mesh editing, Multimedia Tools Appl. 71 (2) (2014) 607622.
[5] V. Kraevoy, A. Sheffer, Mean-value geometry encoding, Int. J. Shape Model. 12 (2006) pp. 29–46.
[6] R.L. Cook, L.C. Carpenter, E.E. Catmull, The reyes image rendering architecture, Proceedings of the 14th annual conference on Computer graphics and interactive techniques (1987).
[7] G. Elber, Geometric texture modeling, IEEE Comput Graph Appl 25 (4) (2005) 66–76.
[8] L. Szirmay-Kalos, T. Umenhoffer, Displacement mapping on the GPU state of the art, Comput. Graphics Forum 27 (6) (2008) 1567–1592.
[9] M. Nießner, C. Loop, Analytic displacement mapping using hardware tessellation, ACM Trans. Graph. 32 (3) (2013).
[10] J.-H. Park, J.-H. Moon, S. Park, S.-H. Yoon, Geostamp: detail transfer based on mean curvature field, Mathematics 10 (3) (2022).
[11] Y.-K. Lai, S.-M. Hu, D.X. Gu, R.R. Martin, et al., Geometric texture synthesis and transfer via geometry images, in: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling, in: SPM '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 1526.
[12] G. Elber, Geometric deformation-displacement maps, in: 10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings., 2002, pp. 156–165.
[13] M. Eyiyurekli, D.E. Breen, Detail-preserving level set surface editing and geometric texture transfer, Graph Models 93 (2017) 39–52.
[14] D. Zhang, X. Wang, J. Hu, H. Qin, Interactive modeling of complex geometric details based on empirical mode decomposition for multi-scale 3D shapes, Comput.-Aided Des. 87 (2017) pp. 1–10.
[15] F. Maggioli, S. Melzi, M. Ovsjanikov, M.M. Bronstein, E. Rodolá, Orthogonalized fourier polynomials for signal approximation and transfer, Comput. Graphics Forum 40 (2) (2021) 435–447.
[16] A. Hertz, R. Hanocka, R. Giryes, D. Cohen-Or, Deep geometric texture synthesis, ACM Trans. Graph. 39 (4) (2020).
[17] W. Yifan, L. Rahmann, O. Sorkine-Hornung, Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields, Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)
[18] D. Boscaini, D. Eynard, D. Kourounis, M.M. Bronstein, Shape-from-operator: recovering shapes from intrinsic operators, Comput. Graphics Forum 34 (2) (2015) pp. 265–274.
[19] H. Zhang, O. Van Kaick, R. Dyer, Spectral mesh processing, Comput. Graphics Forum 29 (6) (2010) 1865–1894.
[20] M. Huska, D. Lazzaro, S. Morigi, Shape partitioning via L_p compressed modes, J. Math. Imaging Vis. 60 (7) (2018) 1111–1131.
[21] S. Morigi, M. Huska, Sparsity-inducing variational shape partitioning, Electron. Trans. Numer. Anal. 46 (2017) 36–54.
[22] Z. Karni, C. Gotsman, Spectral compression of mesh geometry, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, in: SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., USA, 2000, p. 279286.
[23] V. Andersen, M. Desbrun, J.A. Bærentzen, H. Aanæs, Height and tilt geometric texture, in: G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, J.-X. Wang, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M.L. Encarnação, C.T. Silva, D. Coming (Eds.), Advances in Visual Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 656–667.
[24] M. Meyer, M. Desbrun, P. Schröder, A. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: Visualization and Mathematics III., 2003, pp. 35–57.
[25] M.S. Floater, Mean value coordinates, Comput Aided Geom Des 20 (1) (2003) 19–27.
[26] A. Sheffer, V. Kraevoy, Pyramid coordinates for morphing and deformation, in: Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004., 2004, pp. 68–75.
[27] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, H.P. Seidel, Differential coordinates for interactive mesh editing, in: Proceedings Shape Modeling Applications, 2004., 2004, pp. 181–190.