Lagrangian Evolution Approach to Surface-Patch Quadrangulation

LAGRANGIAN EVOLUTION APPROACH TO SURFACE-PATCH QUADRANGULATION

Martin Húska, Bologna, Matej Medľa, Bratislava,
Karol Mikula, Bratislava, Serena Morigi, Bologna

(Received December 17, 2019)

Abstract. We present a method for the generation of a pure quad mesh approximating a discrete manifold of arbitrary topology that preserves the patch layout characterizing the intrinsic object structure. A three-step procedure constitutes the core of our approach which first extracts the patch layout of the object by a topological partitioning of the digital shape, then computes the minimal surface given by the boundaries of the patch layout (basic quad layout) and then evolves it towards the object boundaries. The Lagrangian evolution of the initial surface (basic quad layout) in the direction of the gradient of the signed distance function is smoothed by a mean curvature term. The direct control over the global quality of the generated quad mesh is provided by two types of tangential redistributions: area-based, to equally distribute the size of the quads, and angle-based, to preserve quad corner angles. Experimental results showed that the proposed method generates pure quad meshes of arbitrary topology objects, composed of well-shaped evenly distributed elements with few extraordinary vertices.

MSC 2020: 35K55, 35K93, 65M08

Keywords: Lagrangian evolution; patch layout; non-conforming mesh; mesh partitioning

1. Introduction

Quad meshes, i.e., meshes made entirely of quadrilaterals, have been widely used for many years to represent 3D models in applications like, e.g., CAD/CAM, computer graphics and scientific computing, because a number of tasks are better suited to quad meshes than to triangular meshes. In these applications, a *patch layout* of the 3D model, defined as a partition of the 2-manifold surface into non-overlapping

This work has been supported by Grants APVV-15-0522, VEGA 1/0608/15, and by the "National Group for Scientific Computation (GNCS-INDAM)".

patches, is highly desirable to support standard operations like e.g. texturing, non-uniform rational basis spline (NURBS) fitting or adaptive simulations.

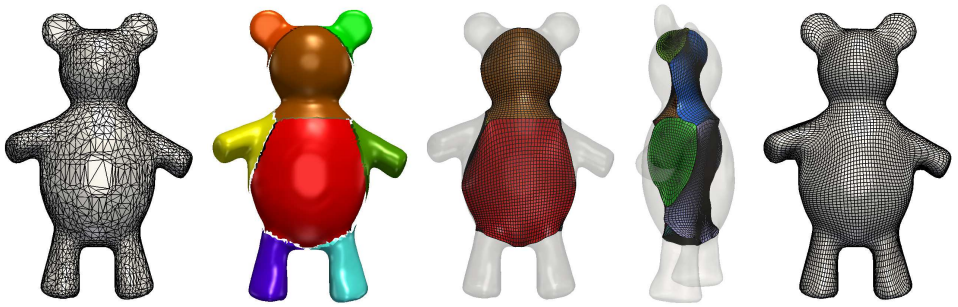
A patch layout easily encodes the underlying high-level geometric and topological structure, thus allowing an easy processing of the underlying surface in a well structured manner. Tools in solid shape modelling supported by boundary representations (B-Reps) strongly rely on the patch layout of 2-manifold surfaces, which, in this case, is represented by a partition of the 2-manifold into non-overlapping, in general 0-genus, n-side patches such that any two patches share an edge, part of an edge, a vertex or are connected by a chain of adjacent patches. Tensor-product NURBS surfaces and T-splines, naturally associated to quad meshes, are preferred for the patch representation being the dominant industry standards.

In general, a patch layout supports the representation of an arbitrary topology object with a pure quad mesh. However, since regular quad meshes are limited to represent surfaces of disk or toroidal topology, one must necessarily resort to semi-regular quad meshes to cover an arbitrary topology manifold. A semi-regular quad mesh is defined by stitching, in a conforming way, several regular 2D arrays of quads side to side. In a semi-regular quad mesh, all vertices that are internal to patches or lie along their boundary edges are regular, while only vertices that lie at corners of patches may possibly be extraordinary [37], [29].

A consistent amount of work has pursued the goal of creating semi-regular quad meshes by constructing an ad hoc 4-side patch layout which can easily ensure the stitching side by side of the generated patches [7], [16], [37]. However, the satisfaction of the conforming constrain leads inevitably towards patch layouts which yield a very large number of patches and are not anymore representative of the high-level structural information which instead has to strictly meet the application needs. Allowing for non-conforming n-side patch layout, for which the intersection of two patches may not be the whole edge or vertex, but a part of an edge, offers higher flexibility in reduction of the number of patches and better representativeness of the object structures. However, at the same time, this introduces an intrinsic difficulty in the generation of 4-side high-quality quad sub-meshes associated to each patch, while consistently stitching them together.

Given an unstructured triangular mesh $M_{\triangle}$, acquired for example from a scanning process of an arbitrary topology object, and an n-side, possibly non-conforming, patch layout characterizing the intrinsic high-level object structure, in this work we propose an automated method for the generation of a pure high-quality quad mesh approximating the discrete manifold $M_{\triangle}$ that preserves the provided n-side non-conforming patch layout, while maintaining the desirable feature of a regular quadrangulation inside the patches.

Traditionally, quad mesh processing algorithms were focused on generating mesh elements with optimal shape and regularity. The most desirable properties pursued in generating quadrilateral meshes usually are: a minimal number of extraordinary vertices (EV) (internal vertices that do not have exactly four neighbours); high-quality elements, i.e., quadrilaterals as close to squares or rectangles as possible; structural information, i.e., the alignment of the elements to mesh features like edges and corners. A discussion of specific characteristics of quad meshes, as well as a survey of recent research on quad mesh processing is provided in [2], [4].



(a) $M_{\triangle}$ (b) $\bigcup_i M_i$ (c) $\bigcup_i S_i$ (d) $M_{\square}$

Figure 1. The fundamental steps of the proposed method: (a) Input mesh $M_{\triangle}$; (b) Patch layout; (c) Basic Quad Layout S from two different camera views; (d) Resulting pure quad mesh $M_{\square}$.

In our approach, the resulting mesh $M_{\square}$ has the following properties:

- ▷ $M_{\square}$ is a non-conformingly semi-regular mesh that accurately represents an arbitrary topology 3D shape by a small number of regular sub-meshes, obtained by sampling and evolving a specific non-conforming patch layout that well represents the salient features of the shape;
- ▷ $M_{\square}$ is composed of well-shaped quad mesh elements which, in terms of edges and angles, are as close to squares or rectangles as possible, distributed with uniform area density.

The overall process consists of three main phases summarized in Algorithm `Surface-Patch Quadrangulation` and illustrated in Figure 1.

Algorithm: `Surface-Patch Quadrangulation`

Input: Triangular Mesh $M_{\triangle}$, required average quad edge length h

Output: Pure Quadrilateral Mesh $M_{\square}$

1: $\{M_i\}_{i=1}^{K} \longleftarrow$ `Mesh_Partitioning`$(M_{\triangle})$

2: $\{S_i\}_{i=1}^{K} \longleftarrow$ `Basic_Quad_Layout`$(\{M_i\}_{i=1}^{K}, h)$

3: $M_{\square} \longleftarrow$ `Basic_Quad_Layout_evolution`$(\{S_i\}_{i=1}^{K}, \{M_i\}_{i=1}^{K})$

Given an input surface, which is represented by an unstructured triangular mesh $M_\triangle$, see Figure 1(a), we first extract its patch layout in Phase 1 by partitioning it into K patches M_i of 0-genus having one boundary, see Figure 1(b). From the given partitioning (chartification), in Phase 2, we describe a new procedure to generate the *Basic Quad Layout* S. We define the *Basic Quad Layout* as the minimal surface version of the underlying shape, where the surface pieces are joined together and represented by quad meshes, according to the boundary of the given patch layout. The Basic Quad Layout structure S of mesh $M_\triangle$ consists of the union of K surfaces S_i, each discretized by a quad grid, according to a given desired edge length h that matches the boundary of the patch, see Figure 1(c). The Basic Quad Layout S is then evolved in Phase 3 towards the input triangulation $M_\triangle$ to create a pure quadrilateral mesh $M_\square$ which well approximates the given 3D shape, see Figure 1(d).

In the present work, both Phase 1 and 2 could admit as input only the cloud of points/vertices on $M_\triangle$, however, the connectivity information is used in the last Phase 3, the Basic Quad Layout Evolution.

The Lagrangian evolution of the Basic Quad Layout is a novel approach to the quad mesh generation. The proposed evolution model follows the direction of the gradient of a signed distance function smoothed by a mean curvature term. The direct control over the global quality of the generated quad mesh is provided by two types of tangential redistributions: area-based, to equally distribute the size of the quads, and angle-based, to preserve quad corner angles.

The rest of the paper is organized as follows. In Section 2, previous work on quad mesh processing and surface evolution is discussed. In Section 3, the chartification method used in the mesh partitioning step is briefly presented. Section 4 describes the new procedure to generate the Basic Quad Layout, while in Section 5 we introduce the proposed evolution framework by a suitable Lagrangian evolution model. Besides the mathematical foundations which are partially described in the Appendix, we provide a description of an efficient implementation based on finite volume discretization. Finally, Section 6 presents some examples and performance evaluations, and we conclude the paper in Section 7 by discussing potential improvements of the method, extensions and future research directions.

2. Related work

2.1. Quad mesh generation. Converting an unstructured mesh into a more regular and structured model is a challenging problem, which has attracted a considerable amount of research in recent years. The literature is quite extensive, we refer the reader to [2], [4] for a general and exhaustive overview of quadrangulation

methods. We survey in the following only the most closely related works which can be categorized as *global mesh optimization* and *patch-layout-based* techniques.

Traditional techniques in *global mesh optimization* include the Direct Tri to Quad conversion and Voronoi-based methods. The first targets directly the problem of converting triangular (or polygonal) input mesh into a quadrangular one utilizing local connectivity operations, e.g., gluing two adjacent triangles into one quad, which makes it highly input-dependent and in general it produces an unstructured quad mesh [35]. Voronoi-based methods use Centroidal Voronoi Tessellation (CVT) to optimize a sampling quality measure. CVTs can be used to generate quad (or quad-dominant) meshes as in [39], [21]. More recent global mesh optimization methods explicitly control local properties of quad elements in the mesh by means of the guiding fields. Usually the quadrangulation process is divided into three steps: Orientation Field Generation (the most popular is the Ro-Sy field [20]), Sizing Field Generation, and Quad Mesh Synthesis. Local solutions proposed in [38], [20] provide favorable CPU times, however, the number of singularities (extraordinary vertices) is high and they in general produce only quad dominant or valence semi-regular quad meshes.

Patch-layout-based methods rely on the construction of a one-to-one mapping of the original surface onto a set of n-side patches. Afterwards, the final quadrangulation is obtained by sampling each patch by a regular grid. State-of-the-art methods use 4-side patches which naturally lead to a conforming patch layout and semi-regular pure-quad mesh.

Recently, a novel approach to quad layout and mesh generation was introduced driven by a carefully defined curve skeleton of the shape (via Reeb Graphs or constraint Laplacian smoothing) that captures the intrinsic features of the processed shape. Then the skeleton information is used in the construction of quad layout and direct quadrilateral mesh generation, see [37].

In [7], [16] the set of quad charts of the global chartification of the input mesh is obtained by using the Morse-Smale Complex of Laplacian Eigenfunctions. Creating a global quadrilateral charting is equivalent to determining the singularity points. A user-assisted method to interactively localize the singularity points is proposed in [36], while in [3] a mixed-integer solver is developed to automatically identify these points.

The proposed surface-patch quadrangulation belongs to the patch-layout-based class of methods where the patch layout is obtained similarly to [7] driven by compact support quasi-eigenfunctions of the Laplace-Beltrami operator [17], thus embedding the high-level object structural information.

However, by relaxing the conformity constrain as in [29], we obtain an n-side patch layout, with $n \geqslant 4$, which offers more flexibility for reduction of the number of patches

while maintaining the representativeness of the salient parts of the object. Finally, the Lagrangian surface evolution distinguishes our quadrangulation proposal from the other parametrization-based representatives of this class and results in regular pure-quad grids for each patch oriented to curvature directions.

2.2. Lagrangian surface evolution by partial differential equations. There are two main techniques dealing with the modelling of surface evolution, the Eulerian and the Lagrangian approaches. Among the Eulerian methods the level-set approach is probably the one mostly used [32], [31], although the so-called phase-field methods are used as well. The level-set methods are utilized in various contexts, ranging from purely physical applications, e.g., front dynamics, to the image processing and computer vision applications such as image segmentation or 3D point cloud surface reconstruction, see e.g. [32], [11]. Since the sampled vertices of the input surface can be understood as a 3D point cloud, in our work we have been inspired by the work by Zhao et al. [42], which, to our knowledge, represents the first level-set approach to the point cloud reconstruction and thus an implicit surface finding and representation.

However, the level-set methods evolve the surface implicitly as an isosurface of a three-dimensional level-set function. The corresponding partial differential equations (PDEs) solved numerically are spatially three-dimensional which can make the numerical solution computationally heavy, especially due to the fact that the final surface reconstruction is obtained as a steady state of the level-set function evolution. With this respect, we mention the work [19] which introduces a computationally efficient semi-implicit method to Zhao et al.'s surface reconstruction approach. Another inherent property of the level-set method is that one has to create an explicit discrete surface representation from its implicit form.

In order to avoid this task and to deal numerically with spatially just 2D problems, the Lagrangian approaches arise naturally. They evolve the surface explicitly, usually approximated by triangular or quadrilateral meshes. A Lagrangian method for 3D point cloud surface reconstruction by directly evolving a triangular mesh was suggested in [6].

In general, any surface evolution can be split into its normal and tangential directions. The normal component of the motion provides the overall image of the surface evolution. The tangential motion is used to stabilize the Lagrangian computational methods but it also allows to control the quality of triangular or quadrilateral surface discretization. The important role of the tangential redistribution of points on evolving curves and surfaces has been emphasized in many publications since the mid-nineties, see e.g. [15], [18], [26], [27], [33], [14] for 2D and 3D curve evolutions, and [1], [28], [25], [6], [10], [23] for surface evolutions. From the point of view of spatial discretizations, two main numerical approaches are used for the evolution

of triangular and quadrilateral surface meshes, the finite volume method [24], [26] leading in the basic setting to the well-known cotangent scheme, and finite element methods introduced by Dziuk for surfaces evolving by the mean curvature in 1991 [8] and then followed by [1], [10] etc.

As noted above, the tangential movement of surface discretization points is necessary for stability of the numerical computations—it prevents self-intersections of the surface and the shape degeneration of finite elements or finite volumes that may arise from a naive numerical approximation. The uniform [26] or curvature-dependent [23] finite volume area redistribution can be successfully performed and a control of the angle redistribution in triangular and quadrilateral meshes is also highly desirable. In the present work we use the finite volume spatial approximation for evolving surfaces represented by a quad mesh and the surface time evolution is accompanied by a suitable area and angle redistribution. Together with the stability of computations, this tangential redistribution enables the control over quad shapes already during the evolution and at the final stage, thus allowing us to obtain high-quality quad mesh surfaces.

3. Phase 1: Mesh partitioning

Let us consider a triangle mesh $M_\triangle = (V, T)$, representing a piecewise linear approximation of a 2-manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$, where $V \in \mathbb{R}^{n_V \times 3}$ is a set of n_V vertices and $T \in \mathbb{N}^{n_T \times 3}$ is a set of n_T triangles that define the connectivity of the mesh.

The partitioning of $M_\triangle$ is the decomposition of $M_\triangle$ into K non-overlapping open sub-meshes (patches) M_i of 0-genus with one boundary connected component such that $M_\triangle = \bigcup_{i=1}^{K} M_i$. Each sub-mesh M_i is characterized by a triplet $M_i = (V_i, T_i, B_i)$, where V_i is a set of vertices, T_i a set of triangles and B_i is a list of boundary vertices in V_i.

Recently in [17] an efficient strategy to obtain such mesh decomposition has been proposed. The partitioning relies on the so-called L_p Compressed Modes that is a sparser version of the Compressed Manifold Modes [30]. The approach iteratively decomposes the mesh into K sub-meshes. The resulting patches define the patch layout of $\mathcal{M}$ and represent the salient parts of $\mathcal{M}$ with the minimum number of patches K satisfying the required genus and boundary criteria.

Without loss of generality, in this work we utilize in Phase 1 the partitioning method introduced in [17]. However, Algorithm `Surface-Patch Quadrangulation` can be easily adapted also to different patch layouts, e.g. [7], [16], [37].

4. Phase 2: Build of the basic quad layout

Given a patch layout of $\mathcal{M}$, we create an associated Basic Quad Layout $S = \bigcup_{i=1}^{K} S_i$, where each sub-part S_i is a surface with boundary curves approximating B_i, thus matching C^0 the common boundary between S_i and S_j, see Figure 1(b)–(c). The benefit of S will be revealed later in the surface approximation by a quadrilateral mesh. However, the layout itself can represent a sort of topology signature of the object $\mathcal{M}$ that can be used in different contexts, e.g., in the generation of the so-called B-Reps in solid modelling.

The overall procedure to fulfill this aim consists of the following three simple steps for each S_i, $i = 1, \ldots, K$:

(1) Construction of the boundary ∂S_i of S_i;
(2) Resolution settings of the boundary of the quadrilateral patch S_i;
(3) Construction of the minimal surface S_i for the given ∂S_i.

The result of this phase is the Basic Quad Layout S represented by K surface patches S_i, each associated to its corresponding M_i. As we will describe later in this section, the use of the Basic Quad Layout S will lead to quadrilateral representation $M_\square$ of $\mathcal{M}$ composed of a number of regular grids that goes from a minimum of K up to a maximum of $5K$.

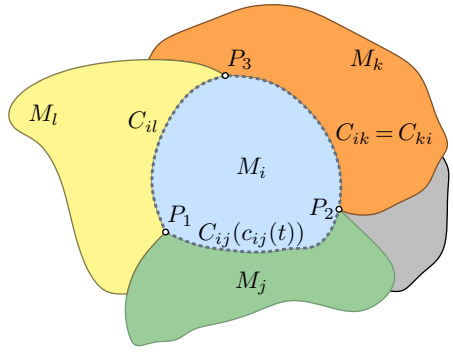


Figure 2. A synthetic scheme of S_i boundary construction.

4.1. Step 1: Construction of the boundary of S_i. The goal of this step is to split the boundaries B_i, $i = 1, \ldots, K$, into a collection of boundary pieces C_{ij} each sharing exactly two sub-meshes M_i and M_j, and approximate them by spline curves.

Figure 2 illustrates a synthetic scheme where the three pieces C_{ij}, C_{ik}, C_{il} for the sub-mesh M_i are marked by dashed red lines.

We first locate a set of vertices $P \subset V$ (marked in blue in Figure 2) that lie on the common boundary of at least three patches of $\mathcal{M}$. Formally, each element of P

is defined as $B_i \bigcap B_j \bigcap B_k$, $i \neq j \neq k$, where the triplet $\{i, j, k\}$ is taken as all the unique combinations out of $\{1, \ldots, K\}$. The vertices in P define the locations for the split of each B_i, $i = 1, \ldots, K$.

Once the set P is selected, we proceed by looping through each patch boundary B_i and by splitting it into pieces C_{ij}, which start and end at elements of P, forming the set C, which reads

$$C := \{C_{ij}\,;\; C_{ij} = B_i \cap B_j \;\forall\,(i,j)\}.$$

We notice that $C_{ij} = C_{ji}$ for all (i, j).

In the case a boundary B_i does not contain any vertex of P, then $C_{ij} \equiv B_i$, i.e., it represents the whole boundary between the two patches M_i and M_j.

Finally, we create an index list I_i for each B_i of the references to elements of C such that $B_i = \bigcup_{j \in I_i} C_{ij}$. In Figure 2 we have $I_i = (j, k, l)$.

At last, we represent each boundary piece C_{ij} by a least square approximating cubic spline $c_{ij}(t)$, $t \in [0, 1]$, such that the spline is clamped at the first and the last point of C_{ij}.

Therefore, the boundary of each S_i is defined as

$$\partial S_i(t) = \bigcup_{j \in I_i} c_{ij}(t). \tag{4.1}$$

4.2. Step 2: Resolution settings of the boundary of the quadrilateral patch S_i. Preliminarily to Step 3, where each S_i will be constructed, we prepare the boundary of the quadrilateral patch S_i by the following process flow:

2.1. Uniform sampling of $\partial S_i(t)$ into points b_i, setting of the quadrilateral grid resolution, placement of the four corner vertices.

2.2. Classification of S_i as *flat* or *protrusion* patch.

4.2.1. Sampling of $\partial S_i(t)$ and resolution setting. Given the desired edge length h for the quads in $M_\square$, each patch boundary ∂S_i is uniformly discretized by a list of boundary points b_i obtained as

$$b_i = \bigcup_{j \in I_i} \Big\{ c_{ij}(t_k)\,;\; t_k \in [0,1],\; k = 1, \ldots, \Big[\frac{|C_{ij}|}{h}\Big] \Big\}, \tag{4.2}$$

where $[\cdot]$ represents the rounding to the closest even number and $|C_{ij}|$ denotes the chord length of a piecewise linear curve computed as

$$|C_{ij}| = \sum_{l=1}^{\#C_{ij}-1} \|C_{ij}^{l+1} - C_{ij}^{l}\|$$

with C_{ij}^{l} and C_{ij}^{l+1} being consecutive points in the list C_{ij}.

Once the boundary list b_i is obtained, we split it into four parts to match the boundary of a quadrilateral grid of resolution (m_i, n_i) that will discretize S_i.

To that aim, we apply the Principal Component Analysis to the vector field of the principal curvature directions at the points V_i of M_i to generate a pair of vectors $\{w_1, w_2\}$ which are used to properly align the four sides of the quadrilateral patch S_i to the main principal curvature directions as this leads to better shaped (e.g., flatter) quads and improves surface approximation.

The grid resolution is set for each S_i as $(m_i = \lfloor \#b_i/4 \rfloor,\ n_i = \lceil \#b_i/4 \rceil)$, and the boundaries of the $m_i \times n_i$ grid which discretizes S_i are placed on the boundary b_i by fixing two adjacent corners, centering them around w_1, such that S_i is well aligned with its neighbouring patches.

Coherent stitching between the adjacent patches S_i and S_j is finally guaranteed by imposing the same points of b_i on the shared boundary.

4.2.2. Classification of S_i. Each patch S_i, $i = 1, \ldots, K$, is classified according to the shape of its corresponding sub-mesh M_i as

- ▷ $M_i^{(1)}$—flat surface,
- ▷ $M_i^{(2)}$—protrusion.

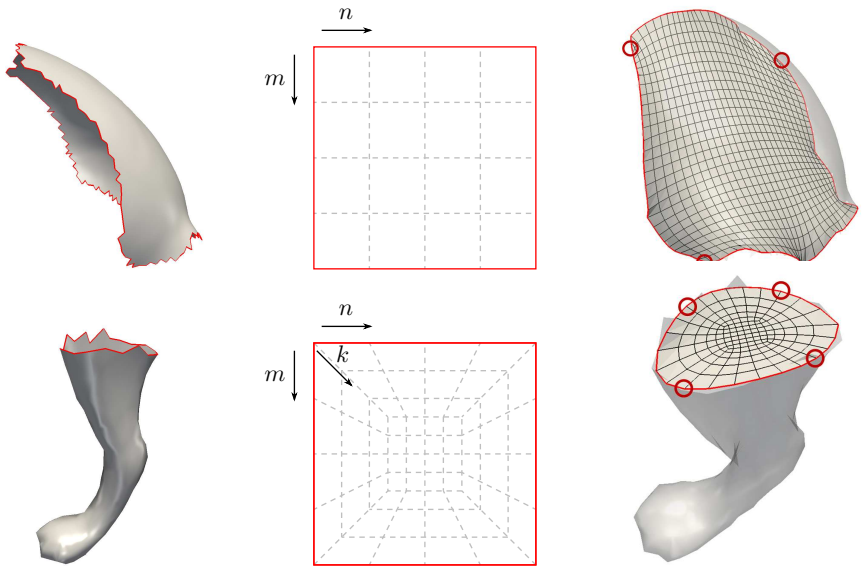


Figure 3. Examples of patches S_i associated to types $M_i^{(1)}$ (top) and $M_i^{(2)}$ (bottom). From left to right: sub-meshes M_i, sketches of their associated grids, patches S_i.

Examples of the patch types $M_i^{(1)}$ and $M_i^{(2)}$ are illustrated in Figure 3 (top-left) and Figure 3 (bottom-left), respectively, together with their associated suitable grids

in Figure 3 (center) used to discretize the corresponding S_i illustrated in Figure 3 (right). The boundaries are highlighted in red and the corners are marked by circles.

The classification is done as follows. For each patch M_i we compute the triangulation area $A_i = \sum_j |T_i^j|$ that we want to cover by quads of size $h \times h$. Thus, the expected number of quads in S_i is approximately given by $q_i^e = A_i/h^2$.

For a flat surface $M_i^{(1)}$ the regular grid discretization of S_i requires the number of quads $q_i^{(1)} = m_i \times n_i$. If $q_i^{(1)} \approx q_i^e$, then S_i is a flat patch, otherwise M_i represents a protrusion patch $M_i^{(2)}$ and the associated S_i patch is discretized by a grid with additional k_i layers forming $q_i^{(2)} = m_i \times n_i + 2k_i(m_i + n_i)$ quads, see Figure 3 (bottom). The number k_i is chosen as the smallest value satisfying $q_i^{(2)} \approx q_i^e$.

In step 3 described in Section 4.3, each patch S_i will be discretized into a quadrilateral grid according to its M_i type, as illustrated in Figure 3 (right). This classification will allow us to preserve uniform discretization following the natural shape of each sub-part M_i.

4.3. Step 3: Construction of the Basic Quad Layout S. The Basic Quad Layout is the union of the surfaces S_i, $i = 1, \ldots, K$, with prescribed boundary ∂S_i, which are constructed by solving on the spatial domain $\Omega \subset \mathbb{R}^3$ the following boundary value problem:

$$\begin{cases} \Delta_{\mathbf{x}} S_i = 0, \\ \partial S_i = b_i. \end{cases} \tag{4.3}$$

The spatial domain Ω corresponds to one of the two domains illustrated in Figure 3 and it is discretized by nodal points $\mathbf{x}_j, j = 1, \ldots, n$, in grids of resolution (m_i, n_i) or (m_i, n_i, k_i) according to the S_i's patch type $M_i^{(1)}$ or $M_i^{(2)}$.

The discretization of the Laplace-Beltrami differential operator (LBO) in (4.3) is based on the uniformly weighted umbrella scheme which uses the grid connectivity information, see Figure 3, on a locally triangulated neighbourhood of each vertex $\mathbf{x}_j \in S_i$ [22]. Thus, the boundary value problem (4.3) leads to the solution of three linear systems $LX = B$, where the system matrix L represents the discretization of LBO and it is highly sparse, diagonally dominant, positive semi-definite, while X are the three coordinates for the grid points $\mathbf{x}_j$ in S_i to be computed. The linear systems can be efficiently solved by the preconditioned Conjugate Gradient method.

This construction guarantees that two neighbouring patches S_i and S_j are consistently stitched together.

In addition, Phase 3 described in Section 5 will allow for the following two evolution strategies: evolution of each S_i separately by fixing its boundary, or, alternatively, evolution of the Basic Quad Layout S as a whole.

We implemented the second strategy more efficiently by first evolving each S_i separately to create an accurate approximation to each salient part M_i, and then jointly moving the surface S, mainly in the tangential direction, to improve the quality of the final quadrilateral mesh.

We finally remark that the constructed Basic Quad Layout S is a non-conformingly pure quad semi-regular mesh which has at most $\#EV = \#P+4K$ extraordinary vertices, and this number will not increase any more during the final evolution Phase 3.

5. Phase 3: Basic quad layout evolution

In this section we describe the evolution of the Basic Quad Layout S towards points on $\mathcal{M}$ approximated by points on $M_\triangle$, to obtain a pure quadrilateral approximation $M_\square$ to the given shape.

To that aim we apply Lagrangian-type evolution with area- and angle-oriented tangential redistribution.

5.1. Lagrangian evolution model. Let us consider a family of parametric surfaces $\{\mathbf{x}(t,u,v)\,;\ (u,v)\in[0,1]^2,\ t\in[0,t_{\text{end}}]\}$ obtained by evolving in time t an initial surface $\mathbf{x}(0,\cdot,\cdot)=S$, where S is the Basic Quad Layout or one of its patches S_i.

The evolution is driven by the following partial differential equation:

$$\begin{aligned}\frac{\partial \mathbf{x}}{\partial t} &= \mathbf{V_N}+\mathbf{V_T}=\beta\mathbf{N}+\mathbf{V_T},\\ \mathbf{x}(0,\cdot,\cdot) &= S,\end{aligned} \tag{5.1}$$

where $\mathbf{V_T}$ is the evolution in tangential direction along the surface defined as

$$\mathbf{V_T}(t,u,v)=\alpha(t,u,v)\mathbf{x}_u+\lambda(t,u,v)\mathbf{x}_v, \tag{5.2}$$

with

$$\mathbf{x}_u=\frac{\partial \mathbf{x}(t,u,v)}{\partial u},\quad \mathbf{x}_v=\frac{\partial \mathbf{x}(t,u,v)}{\partial v} \tag{5.3}$$

representing the partial derivatives of $\mathbf{x}(t,u,v)$, which span the tangent plane; and $\mathbf{V_N}$ represents the evolution by speed β in the outward unit normal direction $\mathbf{N}$ to the surface computed as

$$\mathbf{N}=\frac{\mathbf{x}_u\times\mathbf{x}_v}{|\mathbf{x}_u\times\mathbf{x}_v|}. \tag{5.4}$$

The outward unit normal $\mathbf{N}$ is considered with respect to the whole Basic Quad Layout S also in the case when we evolve only a patch S_i.

The component in the normal direction $\mathbf{V_N}$ affects the surface image, while, at least in the continuous setting, $\mathbf{V_T}$ has no impact on the surface image. However, as shown in literature, see e.g. [25], this kind of tangential movement is very useful especially in the case of Lagrangian-type evolutions. At a specific time t, the result of (5.1) is a 2-manifold $\mathcal{M}(t) = \mathbf{x}(t, \cdot, \cdot)$ and at the final time $\mathbf{x}(t_{\text{end}}, \cdot, \cdot) = M_{\square}$, the desired quadrilateral approximation to the given shape $M_{\triangle}$.

5.1.1. Evolution in the normal direction. Our aim is to prescribe the normal-direction evolution $\mathbf{V_N}$ in (5.1) in such a way that the evolving surface will be moving towards the given shape. An intuitive way would be to set an advection in the normal direction

$$\mathbf{V_N} = -(\nabla |d(\mathbf{x})| \cdot \mathbf{N})\mathbf{N}, \tag{5.5}$$

where $d(\mathbf{x})$ represents the signed distance function to the given shape [41]. The signed distance function is defined to be positive inside the overall Basic Quad Layout S and negative outside. Let us notice that in our definition $|d(\mathbf{x})|$ represents the distance function in the usual sense.

Since in general the distance function $d(\mathbf{x})$ is non-smooth, a smooth evolution is achieved by adding an additional term depending on the mean curvature H of $\mathcal{M}$, thus obtaining

$$\mathbf{V_N} = H\mathbf{N} - (\nabla |d(\mathbf{x})| \cdot \mathbf{N})\mathbf{N}, \tag{5.6}$$

where the mean curvature vector field in (5.6) can be expressed in terms of the Laplace-Beltrami operator as

$$H\mathbf{N} = \Delta_{\mathbf{x}}\mathbf{x} = \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}}\mathbf{x}. \tag{5.7}$$

Let us notice that in (5.7) we understand the mean curvature as a sum of principal curvatures instead of their average.

The divergence operator of a vector field $\mathbf{w} = A\mathbf{x}_u + \Lambda\mathbf{x}_v$, lying in the tangent space of $\mathcal{M}$, is defined following [12] as

$$\nabla_{\mathbf{x}} \cdot \mathbf{w} = \nabla_{\mathbf{x}} \cdot (A\mathbf{x}_u + \Lambda\mathbf{x}_v) = \frac{1}{g}((gA)_u + (g\Lambda)_v), \tag{5.8}$$

where g (see (5.16)) is a scalar field on $\mathcal{M}$ related to the surface parametrization; and the intrinsic gradient of a generic scalar function $\varphi(\cdot, \cdot)$ is defined by [12] as

$$\nabla_{\mathbf{x}}\varphi = \frac{(\mathbf{x}_v \cdot \mathbf{x}_v)\varphi_u - (\mathbf{x}_u \cdot \mathbf{x}_v)\varphi_v}{g^2}\mathbf{x}_u + \frac{(\mathbf{x}_u \cdot \mathbf{x}_u)\varphi_v - (\mathbf{x}_u \cdot \mathbf{x}_v)\varphi_u}{g^2}\mathbf{x}_v. \tag{5.9}$$

In order to control the trade-off between the advection and diffusion terms in (5.6), we introduce two functions $\varepsilon(d(\mathbf{x}))$ and $\eta(d(\mathbf{x}))$ depending on the signed distance function $d(\cdot)$ at point $\mathbf{x}$, and modify (5.6) to

$$\mathbf{V}_{\mathbf{N}} = \varepsilon(d(\mathbf{x}))\Delta_{\mathbf{x}}\mathbf{x} + \eta(d(\mathbf{x}))\mathbf{N}. \tag{5.10}$$

The role of the coefficient $\varepsilon(d(\mathbf{x}))$ in the diffusion term is to obtain stronger smoothing of the evolving surface in the case $\mathbf{x}(\cdot)$ is far from $M_{\triangle}$; therefore $\varepsilon(d(\mathbf{x}))$ is defined by

$$\varepsilon(d(\mathbf{x})) = c_1(1 - \mathrm{e}^{-d(\mathbf{x})^2/c_2}), \tag{5.11}$$

where c_1 and c_2 are parameters controlling the shape of $\varepsilon(\cdot)$; in particular c_2 controls the transition width of the function, and c_1 controls its amplitude. In Figure 4 we plot the function $\varepsilon(t)$ for different values of c_2, and $c_1 = 1$.

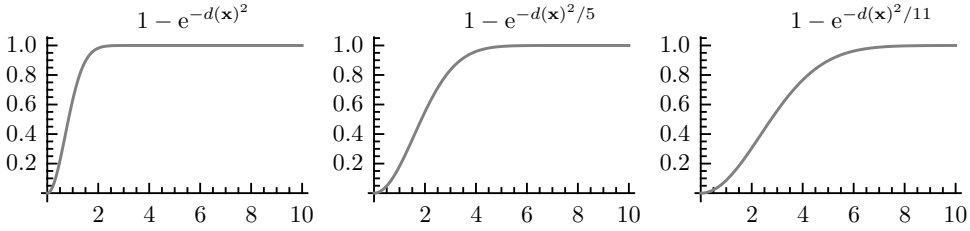


Figure 4. Function $\varepsilon(t)$ with $c_1 = 1$ and varying $c_2 = \{1, 5, 11\}$.

The function $\eta(d(\mathbf{x}))$ in (5.10) is given by

$$\eta(d(\mathbf{x})) = d(\mathbf{x})\big(|-\nabla d(\mathbf{x}) \cdot \mathbf{N}| + \sqrt{1 - (\nabla d(\mathbf{x}) \cdot \mathbf{N})^2}\big). \tag{5.12}$$

The signed distance function $d(\mathbf{x})$ in (5.12) that multiplies the brackets has two purposes. It accelerates the movement if the surface is far away from the triangulation and it flips the movement direction if it is outside of the triangulation. The first term in the parentheses represents the modification of (5.5). This term can be insufficient in the case the normal vector $\mathbf{N}$ and $\nabla d(\mathbf{x})$ are perpendicular. The second term in the parentheses deals with this problem. It is the length of projection of $-\nabla d(\mathbf{x})$ onto the tangential plane. As a consequence it diminishes when $-\nabla d(\mathbf{x})$ is parallel to the normal vector of the surface $\mathbf{N}$ and if $-\nabla d(\mathbf{x}) \perp \mathbf{N}$, then $\eta(d(\mathbf{x}))$ is proportional to the distance $d(\mathbf{x})$.

An insight into $\eta(\cdot)$ is sketched in Figure 5 in which a surface patch S_i (drawn in solid green) evolves towards the triangulation patch M_i of a hat object (drawn in solid red). At time step $t = 0$, see the bottom part of Figure 5, the usual term

$-\nabla|d(\mathbf{x})|\cdot\mathbf{N}$ would indicate to evolve S_i towards a different patch M_j adjacent to M_i. In this case the second term in (5.12) is zero, thus the absolute value $|-\nabla d(\mathbf{x})\cdot\mathbf{N}|$ forces the surface to evolve in the direction $\mathbf{N}$. At time $t=1$ we have $-\nabla d(\mathbf{x})\perp\mathbf{N}$, in this case the first term is zero while the second term results in $d(\mathbf{x})$, thus, forcing the surface to evolve farther in the normal direction. For $t=2$ both terms in (5.12) are positive, dictating the movement along $\mathbf{N}$. At last, at $t=3$ we have a possible overflow of the evolving surface $\mathbf{x}(t,\cdot)$. The second term in (5.12) results in almost zero, while the first part dictates the evolution in the normal direction. However, since $d(\mathbf{x})<0$, the surface $\mathbf{x}(t,\cdot)$ will move back towards M_i.

Summarizing, the evolution in the normal direction in (5.1) is driven by (5.10) consisting of an advection in the normal direction with speed (5.12) and a smoothing diffusion force (5.11) controlling the mean curvature flow of the surface.

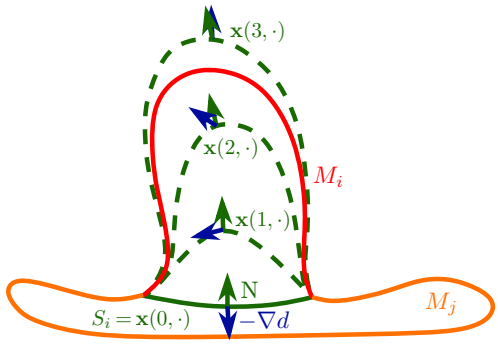


Figure 5. Influence of $\eta(\cdot)$ in (5.12) acting on $\mathbf{x}(t,\cdot)$ (green coloured lines) evolving towards a patch M_i, coloured in solid red. From bottom to top the different configurations of the vectors $\mathbf{N}$ and $-\nabla d$ are shown at a point of the evolving surface $\mathbf{x}(t,\cdot)$ for $t\in\{0,1,2,3\}$.

5.1.2. Redistribution in the tangential direction. The goal to maintain a well-shaped quad mesh during the evolution motivates us to incorporate into the model (5.1) the tangential control of two important properties of the quad mesh: the area size and the angles of the quad elements. To this aim the tangential component in (5.1) is defined as

$$\mathbf{V_T}=\mathbf{V_T^r}+\mathbf{V_T^n}, \tag{5.13}$$

where the vector $\mathbf{V_T^r}$ contributes to the quad size control and the vector $\mathbf{V_T^n}$ handles the control of the angles of the quads. Then for the components α, λ of $\mathbf{V_T}$ in (5.2) it holds that

$$\alpha(u,v)=\alpha^r(u,v)+\alpha^n(u,v),\quad \lambda(u,v)=\lambda^r(u,v)+\lambda^n(u,v). \tag{5.14}$$

In the following, we first describe the angle-based redistribution term $\mathbf{V}_{\mathbf{T}}^n$ in terms of $\mathbf{x}_u$ and $\mathbf{x}_v$ and then we give the background for the area control term $\mathbf{V}_{\mathbf{T}}^r$.

5.1.3. Angle-based redistribution. The purpose of this redistribution is to control the angles of quadrilaterals that are composing the surface. Let us assume that the surface is divided into quadrilateral patches. In the discrete setting these patches are represented by the quads which compose the surface. The angle redistribution is directly dependent on the vectors $\mathbf{x}_u$ and $\mathbf{x}_v$. Let $\mathbf{x}_i$, $i = 1, \ldots, n$, be a corner point of a quadrilateral patch and n be the number of all corner points of quadrilateral patches on the surface. Let $N_\square(\mathbf{x}_i)$ be a set of quadrilateral patches that contain $\mathbf{x}_i$ and let $\#N_\square(\mathbf{x}_i)$ be its cardinality. Let $\mathbf{x}_u^j, \mathbf{x}_v^j$ be the tangent vectors of the jth neighbouring quadrilateral patch.

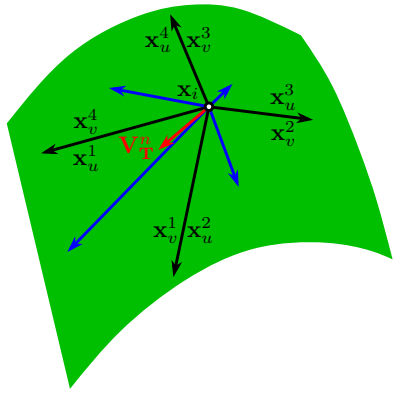


Figure 6. A scheme illustrating the construction of the vector $\mathbf{V}_{\mathbf{T}}^n$ (red colour) in a local neighbourhood of a surface point $\mathbf{x}_i$.

At each corner point $\mathbf{x}_i$, $i = 1, \ldots, n$, the angle-based tangential velocity $\mathbf{V}_{\mathbf{T}}^n$ is constructed as

$$\mathbf{V}_{\mathbf{T}}^n = \mathrm{proj}_{\mathbf{T}}\left(\frac{\omega_n}{\#N_\square(\mathbf{x}_i)} \sum_{j \in N_\square(\mathbf{x}_i)} \left(1 + \frac{\mathbf{x}_u^j}{|\mathbf{x}_u^j|} \cdot \frac{\mathbf{x}_v^j}{|\mathbf{x}_v^j|}\right)(\mathbf{x}_u^j + \mathbf{x}_v^j)\right). \tag{5.15}$$

Formula (5.15) utilizes the fact that the cosine of the angle between two vectors can be computed as the inner product $(\mathbf{x}_u^j/|\mathbf{x}_u^j|) \cdot (\mathbf{x}_v^j/|\mathbf{x}_v^j|)$ and such weights are forced to be positive by a simple shift. Intuitively, in the case of an acute angle, moving $\mathbf{x}_i$ in the direction $\mathbf{x}_u^j + \mathbf{x}_v^j$ enlarges the angle between $\mathbf{x}_u^j$ and $\mathbf{x}_v^j$. As the resulting vector does not have to lie in the tangent plane, we project it on the tangent plane utilizing $\mathrm{proj}_{\mathbf{T}}(\mathbf{V}) = \mathbf{V} - (\mathbf{V} \cdot \mathbf{N})\mathbf{N}$. Since each parametrization contributes to the movement of $\mathbf{x}_i$ in a different direction, the expected behaviour will be a compromise among the parametrizations, leading to a mean angle at a vertex along the neighbouring quads

in the discrete setting. Such angle-based tangential movement of corner points is then extended to the inner points of quadrilateral patches, e.g., by a bilinear interpolation. Parameter ω_n controls the strength of the angle redistribution and it is set constant for all $\mathbf{x}_i$.

In Figure 6 we give an example of the construction of the vector $\mathbf{V}_\mathbf{T}^n$ in (5.15) for a corner point $\mathbf{x}_i$ with four adjacent quadrilateral patches. The vectors resulting from the inner products between two adjacent quadrilateral patches are highlighted in blue and the resulting tangential vector $\mathbf{V}_\mathbf{T}^n$ is drawn in red.

5.1.4. Area-based redistribution. Unlike the angle-based redistribution term $\mathbf{V}_\mathbf{T}^n$, which directly depends on the parametrization, the area-based redistribution term $\mathbf{V}_\mathbf{T}^r$ depends only on the local area density g of $\mathcal{M}$ defined as

$$g = |\mathbf{x}_u \times \mathbf{x}_v| = \sqrt{\det \begin{bmatrix} \mathbf{x}_u \cdot \mathbf{x}_u & \mathbf{x}_u \cdot \mathbf{x}_v \\ \mathbf{x}_v \cdot \mathbf{x}_u & \mathbf{x}_v \cdot \mathbf{x}_v \end{bmatrix}}. \tag{5.16}$$

We are interested in the evolution of g with respect to the normal and tangential velocities in (5.1). The time evolution of g is described in Lemma 5.1.

Lemma 5.1. *Let $\mathbf{x}(t,\cdot)$ be a smooth parametric surface representing the embedding of an evolving 2-manifold in $\mathbb{R}^3$ in time t according to* (5.1)*. Then the local area density g of $\mathbf{x}(t,\cdot)$ evolves as*

$$g_t = -gH\beta + g\nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}, \tag{5.17}$$

where H is the mean curvature scalar field and $\nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}$ is the surface divergence of the tangential vector field $\mathbf{V}_\mathbf{T}$ defined in (5.8)*.*

For completeness and the readers' convenience we postpone the proof to Appendix (A).

If we want the surface area density g to converge to a prescribed area density c during the evolution, we impose the evolution of g w.r.t. the area A of the surface as

$$\Big(\frac{g}{A}\Big)_t = \Big(\frac{c}{A} - \frac{g}{A}\Big)\omega_r, \tag{5.18}$$

where the scalar parameter ω_r controls the speed of convergence. Let us notice that for $\omega_r = 0$, there is no change in time, thus, the density remains preserved as it was at the beginning of the evolution. Formula (5.18) allows to manipulate the local density g in order to asymptotically reach the prescribed value c as $t \to \infty$.

In our case we want g to converge to a uniform area density distribution. The uniform area density distribution is constant all over the space and each valid area density has to integrate to the area of the surface. For this reason c is given by

$$c = \frac{\int_0^1 \int_0^1 g \,\mathrm{d}u \,\mathrm{d}v}{\int_0^1 \int_0^1 \,\mathrm{d}u \,\mathrm{d}v} = A.$$

Let us suppose that the vectors $\mathbf{V}_\mathbf{T}^n$ and $\beta\mathbf{N}$ are already known. The quad area control will be performed in the discrete setting through the control of the finite volume areas.

A relation between $\mathbf{V_T}$ and g that satisfies (5.18) is summarized in Theorem 5.1 in terms of the divergence operator.

Theorem 5.1. *Let $\mathbf{x}(t,\cdot)$ be a smooth parametric surface representing the embedding of an evolving 2-manifold in $\mathbb{R}^3$ according to* (5.1)*. Then $\mathbf{x}(t,\cdot)$ evolves to a surface with area density c if the divergence of the tangential vector field $\mathbf{V}_\mathbf{T}^r$ satisfies the following relation:*

$$\nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}^r = -\nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}^n + H\beta - \frac{1}{A} \iint_\mathcal{M} H\beta \,\mathrm{d}\mathcal{M} + \Big(\frac{c}{g} - 1\Big)\omega_r. \tag{5.19}$$

The proof of Theorem 5.1 is postponed to Appendix (B). Moreover, in order to move the boundary points only along the boundary $\partial\mathcal{M}$ of an open surface $\mathcal{M}$, we can prescribe the following natural boundary condition:

$$\mathbf{V}_\mathbf{T}^r \cdot \mathbf{n}|_{\partial\mathcal{M}} = 0. \tag{5.20}$$

Corollary 5.1 provides a unique solution to formula (5.19) in Theorem 5.1 in terms of a potential field φ.

Corollary 5.1. *Let us assume that $\mathbf{V}_\mathbf{T}^r$ is a gradient vector field of a potential φ. By fixing the value of φ at one point of the surface $\mathbf{x}$ and imposing the homogeneous Neumann boundary conditions for an open surface, equation* (5.19) *has a unique solution given by solving the following equation:*

$$\Delta_\mathbf{x}\varphi = \nabla_\mathbf{x} \cdot \nabla_\mathbf{x}\varphi = -\nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}^n + H\beta - \frac{1}{A} \iint_\mathcal{M} H\beta \,\mathrm{d}\mathcal{M} + \Big(\frac{c}{g} - 1\Big)\omega_r. \tag{5.21}$$

By imposing only the Neumann boundary conditions for an open surface problem, (5.21) attains infinitely many solutions which differ only by a constant, thus the gradient $\mathbf{V}_\mathbf{T}^r = \nabla_\mathbf{x}\varphi$ is naturally the same for each solution. However, a unique solution for (5.21) can be obtained by imposing a Dirichlet boundary condition at one point of $\mathbf{x}$.

5.2. Numerical scheme. In this section we derive the numerical scheme for the Basic Quad Layout evolution driven by the PDE model (5.1). For the spatial discretization, we adopt the finite volume approach, while a semi-implicit scheme is considered in time to linearize the Laplace-Beltrami operator and other non-linear terms in (5.1).

In the preliminary step, the distance function $d(\mathbf{x})$ to $M_\triangle$ is discretized on a voxelized grid inside the $M_\triangle$'s bounding box. Each voxel is represented by a cube of edge length given by 1% of the longest bounding box edge. The values are approximated using the Fast Sweeping Algorithm [40] and modified to a signed distance function by a simple flooding algorithm, which changes the sign of all external voxels relative to $M_\triangle$.

In what follows, we first introduce the finite volume method (FVM) that is used to spatially discretize the quad-based surface PDE evolution model.

Let us consider a piecewise bilinear discretization of the Basic Quad Layout produced in Phase 2 represented by a quad mesh $M_\square := (\{\mathbf{x}_i\}_{i=1}^n, Q)$, where $\{\mathbf{x}_i\}_{i=1}^n$ is a set of n vertices and Q is the set of quads. Then we denote by $N_\square(\mathbf{x}_i) = \{q \in Q; \mathbf{x}_i \in q\}$ the quad 1-ring neighbourhood and by $\#N_\square(\mathbf{x}_i)$ the number of quads surrounding (sharing) the vertex $\mathbf{x}_i$.

The finite volume method assumes in general that the continuous surface $\mathcal{M}$ is approximated by the union of so-called *control volumes* $\mathcal{V}_i$, $i = 1, \ldots, n$, built around each vertex $\mathbf{x}_i$. In the case of an evolving surface, the control volume $\mathcal{V}_i$ naturally changes with the evolving vertex.

Let us introduce the local vertex and quad indexing in the barycentric control volume $\mathcal{V}_i$ around vertex $\mathbf{x}_i$ as illustrated in Figure 7 for $\#N_\square(\mathbf{x}_i) = 5$. The local vertex indices in a quad Q_j are denoted by $\mathbf{x}_j^k$, $k \in \{0, \ldots, 3\}$ with $\mathbf{x}_j^0 = \mathbf{x}_i$, where the index j refers to the neighbouring quads. Therefore, the barycentric finite volume $\mathcal{V}_i$ around the vertex $\mathbf{x}_i$ consists of $m_i = \#N_\square(\mathbf{x}_i)$ one-quarter-quads of the neighbouring quads Q_j as illustrated in Figure 7. The boundary of $\mathcal{V}_i$ in the one-quarter-quad Q_j, $j \in N_\square(\mathbf{x}_i)$ is prescribed by the vertices $\mathbf{r}_j^k$ given as follows:

$$\begin{aligned}
\mathbf{r}_j^1 &= \frac{1}{2}(\mathbf{x}_j^0 + \mathbf{x}_j^1),\\
\mathbf{r}_j^2 &= \frac{1}{4}(\mathbf{x}_j^0 + \mathbf{x}_j^1 + \mathbf{x}_j^2 + \mathbf{x}_j^3),\\
\mathbf{r}_j^3 &= \frac{1}{2}(\mathbf{x}_j^0 + \mathbf{x}_j^3),
\end{aligned}$$

and the area of the finite volume $\mathcal{V}_i$ is then given by

$$m(\mathcal{V}_i) = \frac{1}{2} \sum_{j \in N_\square(\mathbf{x}_i)} |(\mathbf{r}_j^1 - \mathbf{x}_j^0) \times (\mathbf{r}_j^2 - \mathbf{x}_j^0)| + |(\mathbf{r}_j^2 - \mathbf{x}_j^0) \times (\mathbf{r}_j^3 - \mathbf{x}_j^0)|.$$

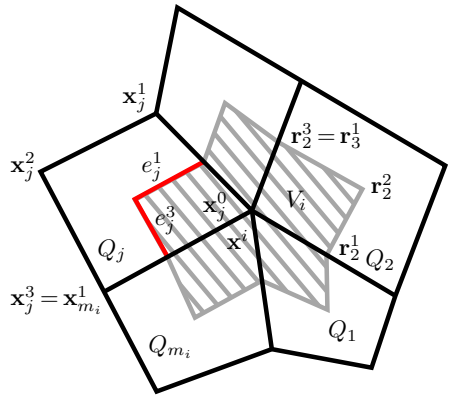


Figure 7. Finite volume $\mathcal{V}_i$ (shaded grey area) with local indexing of quads, corner points and edges.

5.2.1. Finite volume scheme for the evolution model. Unlike the finite differences method, the FVM aims to approximate the solution of an evolutionary PDE model in terms of integrals over the control volumes $\mathcal{V}_i$. Thus we integrate (5.1) over the finite volume $\mathcal{V}_i$, where for $\mathbf{V_N}$ we use (5.10). We get

$$\int_{\mathcal{V}_i} \mathbf{x}_t \,\mathrm{d}\mathcal{M} = \int_{\mathcal{V}_i} \varepsilon(d(\mathbf{x}))\Delta_{\mathbf{x}}\mathbf{x} \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} \eta(d(\mathbf{x}))\mathbf{N} \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^n \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^r \,\mathrm{d}\mathcal{M}. \tag{5.22}$$

Assuming the functions $\varepsilon(\cdot)$ in (5.11) and $\eta(\cdot)$ in (5.12) are given by constant values over the control volume $\mathcal{V}_i$, i.e., $\varepsilon_i = \varepsilon(d(\mathbf{x}_i))$, $\eta_i = \eta(d(\mathbf{x}_i))$, and applying Green's theorem we can rewrite (5.22) to

$$\int_{\mathcal{V}_i} \mathbf{x}_t \,\mathrm{d}\mathcal{M} = \varepsilon_i \int_{\partial\mathcal{V}_i} \nabla_{\mathbf{x}}\mathbf{x} \cdot \mathbf{n}_i \,\mathrm{d}s + \eta_i \int_{\mathcal{V}_i} \mathbf{N} \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^n \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^r \,\mathrm{d}\mathcal{M}. \tag{5.23}$$

The *first* term on the right-hand side of (5.23), using the definition of finite volumes, can be rewritten as

$$\varepsilon_i \int_{\partial\mathcal{V}_i} \nabla_{\mathbf{x}}\mathbf{x} \cdot \mathbf{n}_i \,\mathrm{d}s = \varepsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j} \nabla_{\mathbf{x}}\mathbf{x} \cdot \mathbf{n}_j \,\mathrm{d}s = \varepsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_j} \,\mathrm{d}s, \tag{5.24}$$

where e_j is an edge of the finite volume $\mathcal{V}_i$ in a quad Q_j that consists of two edges e_j^1 and e_j^3, and $\mathbf{n}_j$ denotes the unit normal to e_j. Moreover, e_j^1 and e_j^3 can be parametrized as

$$\begin{aligned} e_j^1(\varrho) &= (\mathbf{r}_j^2 - \mathbf{r}_j^1)\varrho + \mathbf{r}_j^1, \quad \varrho \in [0,1], \\ e_j^3(\phi) &= (\mathbf{r}_j^2 - \mathbf{r}_j^3)\phi + \mathbf{r}_j^3, \quad \phi \in [0,1]. \end{aligned}$$

Thus, we can divide each term of the sum (5.24) into a sum of integrals along e_j^1 and e_j^3, $j = 1, \ldots, \#N_\square(\mathbf{x}_i)$,

$$\varepsilon_i \int_{\partial \mathcal{V}_i} \nabla_{\mathbf{x}} \mathbf{x} \cdot \mathbf{n} \, \mathrm{d}s = \varepsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j^1} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_j^1} \, \mathrm{d}s + \int_{e_j^3} \frac{\partial \mathbf{x}}{\partial \mathbf{n}_j^3} \, \mathrm{d}s. \tag{5.25}$$

The normal derivatives are approximated using bilinear interpolation and the detailed formulas are derived in the Appendix (C). Using formula (7.16) from the Appendix (C), we can approximate (5.25), thus (5.24), as

$$\begin{aligned} &\varepsilon_i \int_{\partial \mathcal{V}_i} \nabla_{\mathbf{x}} \mathbf{x} \cdot \mathbf{n} \, \mathrm{d}s \\ &\quad \approx \varepsilon_i \sum_{j \in N_\square(\mathbf{x}_i)} \frac{m(e_j^1)}{|\mathbf{N}_j^1|} \Big[\frac{1}{4}(-3\mathbf{x}_j^0 + 3\mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2) - \frac{a_j^1}{2}(-\mathbf{x}_j^0 - \mathbf{x}_j^1 + \mathbf{x}_j^3 + \mathbf{x}_j^2) \Big] \\ &\qquad + \frac{m(e_j^3)}{|\mathbf{N}_j^3|} \Big[\frac{1}{4}(-3\mathbf{x}_j^0 - \mathbf{x}_j^1 + 3\mathbf{x}_j^3 + \mathbf{x}_j^2) - \frac{a_j^3}{2}(-\mathbf{x}_j^0 + \mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2) \Big], \end{aligned} \tag{5.26}$$

where $\mathbf{N}_j^1$, $\mathbf{N}_j^3$, a_j^1, a_j^3 are defined in (7.12)–(7.13) and $m(e_j^k) = |\mathbf{r}_j^2 - \mathbf{r}_j^k|$ denotes the Euclidean length of edge e_j^k.

The *second* term on the right-hand side of (5.23) can be rewritten as

$$\eta_i \int_{\mathcal{V}_i} \mathbf{N} \, \mathrm{d}\mathcal{M} = \frac{\eta_i}{2} \sum_{j \in N_\square(\mathbf{x}_i)} (\mathbf{r}_j^1 - \mathbf{x}_j^0) \times (\mathbf{r}_j^2 - \mathbf{x}_j^0) + (\mathbf{r}_j^2 - \mathbf{x}_j^0) \times (\mathbf{r}_j^3 - \mathbf{x}_j^0). \tag{5.27}$$

For the numerical approximation of the *third* term on the right-hand side of (5.23), the integral of $\mathbf{V}_{\mathbf{T}}^n$ defined by equation (5.15), we first approximate the derivatives on the jth quad $\mathbf{x}_u^j$, $\mathbf{x}_v^j$ by

$$\mathbf{x}_u^j \approx \mathbf{x}_j^1 - \mathbf{x}_j^0, \quad \mathbf{x}_v^j \approx \mathbf{x}_j^3 - \mathbf{x}_j^0, \tag{5.28}$$

and then by multiplying it by the area of the finite volume, we get

$$\begin{aligned} \int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^n \, \mathrm{d}\mathcal{M} &\approx \frac{m(\mathcal{V}_i)}{\#N_\square(\mathbf{x}_i)} \\ &\quad \times \sum_{j \in N_\square(\mathbf{x}_i)} \Big(1 + \frac{\mathbf{x}_j^1 - \mathbf{x}_j^0}{|\mathbf{x}_j^1 - \mathbf{x}_j^0|} \cdot \frac{\mathbf{x}_j^3 - \mathbf{x}_j^0}{|\mathbf{x}_j^3 - \mathbf{x}_j^0|} \Big) (\mathbf{x}_j^1 - \mathbf{x}_j^0 + \mathbf{x}_j^3 - \mathbf{x}_j^0). \end{aligned} \tag{5.29}$$

Concerning the *fourth* term of the right-hand side of (5.23), under the assumption that the values of the potential φ are already computed, as described in Section 5.3,

$\mathbf{V}_{\mathbf{T}}^r$ is determined as the surface gradient of φ and it can be computed using the following identity, see e.g. [9],

$$\int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^r \,\mathrm{d}\mathcal{M} = \int_{\mathcal{V}_i} \nabla_{\mathbf{x}}\varphi \,\mathrm{d}\mathcal{M} = \int_{\partial\mathcal{V}_i} \varphi\mathbf{n} \,\mathrm{d}s - \int_{\mathcal{V}_i} \varphi H\mathbf{N} \,\mathrm{d}\mathcal{M}. \tag{5.30}$$

The boundary integral on the right-hand side of (5.30) is approximated using the bilinear interpolation, see Appendix (C),

$$\begin{aligned}\int_{\partial\mathcal{V}_i} \varphi\mathbf{n} \,\mathrm{d}s &= \sum_{j\in N_\square(\mathbf{x}_i)} \int_{e_j^1} \varphi\mathbf{n}_j^1 \,\mathrm{d}s + \int_{e_j^3} \varphi\mathbf{n}_j^3 \,\mathrm{d}s \\ &\approx \sum_{j\in N_\square(\mathbf{x}_i)} \int_{e_j^1} \varphi\frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} \,\mathrm{d}s + \int_{e_j^3} \varphi\frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|} \,\mathrm{d}s \\ &\approx \sum_{j\in N_\square(\mathbf{x}_i)} m(e_j^1)\Big(\frac{3}{8}\varphi_i + \frac{3}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^3\Big)\frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} \\ &\quad + m(e_j^3)\Big(\frac{3}{8}\varphi_i + \frac{3}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^3\Big)\frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|},\end{aligned} \tag{5.31}$$

where φ_j^k is the value of the function φ at point $\mathbf{x}_j^k$.

The second integral term of (5.30) is approximated by taking φ constant on the finite volume and by applying Green's theorem:

$$\begin{aligned}\int_{\mathcal{V}_i} \varphi H\mathbf{N} \,\mathrm{d}\mathcal{M} &= \varphi_i \int_{\mathcal{V}_i} H\mathbf{N} \,\mathrm{d}\mathcal{M} = \varphi_i \int_{\partial\mathcal{V}_i} \frac{\partial\mathbf{x}}{\partial\mathbf{n}_i} \,\mathrm{d}s = \varphi_i \int_{\partial\mathcal{V}_i} \mathbf{n}_i \,\mathrm{d}s \\ &\approx \varphi_i\Big(\sum_{j\in N_\square(\mathbf{x}_i)} \int_{e_j^1} \frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} \,\mathrm{d}s + \int_{e_j^3} \frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|} \,\mathrm{d}s\Big) \\ &\approx \varphi_i\Big(\sum_{j\in N_\square(\mathbf{x}_i)} m(e_j^1)\frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} + m(e_j^3)\frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|}\Big).\end{aligned} \tag{5.32}$$

Putting (5.31) and (5.32) together, the last term on the right-hand side of (5.23) is approximated as

$$\begin{aligned}\int_{\mathcal{V}_i} \mathbf{V}_{\mathbf{T}}^r \,\mathrm{d}\mathcal{M} \approx &\sum_{j\in N_\square(\mathbf{x}_i)} m(e_j^1)\Big(-\frac{5}{8}\varphi_i + \frac{3}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^3\Big)\frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} \\ &+ m(e_j^3)\Big(-\frac{5}{8}\varphi_i + \frac{3}{8}\varphi_j^2 + \frac{1}{8}\varphi_j^1 + \frac{1}{8}\varphi_j^3\Big)\frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|}.\end{aligned} \tag{5.33}$$

Finally, the left-hand side term in (5.23) is approximated by means of the forward finite difference as

$$\int_{\mathcal{V}_i} \mathbf{x}_t \,\mathrm{d}\mathcal{M} = m(\mathcal{V}_i)\Big(\frac{\mathbf{x}_i^{t+\Delta t} - \mathbf{x}_i^t}{\Delta t}\Big). \tag{5.34}$$

Then the approximations (5.26), (5.27), (5.29) and (5.33) are treated in time semi-implicitly, that means the linear terms are assumed in the time step $t + \Delta t$ and the non-linear terms are evaluated in the time step t. Thus, the solution of (5.23) reduces to the following system of linear equations:

$$A_x^t \mathbf{x}^{t+\Delta t} = b_x^t, \tag{5.35}$$

where $\mathbf{x}^{t+\Delta t} = \{\mathbf{x}_1^{t+\Delta t}, \dots, \mathbf{x}_n^{t+\Delta t}\}$. The system matrix A_x^t is sparse and in general non-symmetric, thus the linear system is solved in every time step by BiCGStab linear solver [34] with Incomplete LUT preconditioner (Incomplete LU factorization with dual-threshold strategy).

5.3. Computing the potential field φ. We recall that in order to obtain $\mathbf{V}_\mathbf{T}^r$, or equivalently the fourth term on the right-hand side in (5.23), we first have to compute the potential field φ in (5.21).

Integrating (5.21) over the finite volume $\mathcal{V}_i$ we get

$$\begin{aligned}\int_{\mathcal{V}_i} \Delta_\mathbf{x} \varphi \,\mathrm{d}\mathcal{M} = &- \int_{\mathcal{V}_i} \nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}^n \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} H\beta \,\mathrm{d}\mathcal{M} \\ &- \int_{\mathcal{V}_i} \frac{1}{A} \iint_{\mathcal{M}} H\beta \,\mathrm{d}\mathcal{M} \,\mathrm{d}\mathcal{M} + \int_{\mathcal{V}_i} \Big(\frac{c}{g} - 1\Big)\omega_r \,\mathrm{d}\mathcal{M}.\end{aligned} \tag{5.36}$$

The left-hand side term in (5.36) leads to the same coefficient matrix as (5.26) for $\varepsilon_i = 1$.

The first term on the right-hand side in (5.36) can be approximated utilizing the divergence theorem and equation (5.31) as

$$\begin{aligned}\int_{\mathcal{V}_i} \nabla_\mathbf{x} \cdot \mathbf{V}_\mathbf{T}^n \,\mathrm{d}\mathcal{M} &= \int_{\partial\mathcal{V}_i} \mathbf{V}_\mathbf{T}^n \cdot \mathbf{n} \,\mathrm{d}\mathcal{M} \\ &\approx \sum_{j \in N_\square(\mathbf{x}_i)} m(e_j^1)\frac{1}{8}(3\mathbf{V}_\mathbf{T}^{n,0} + 3\mathbf{V}_\mathbf{T}^{n,1} + \mathbf{V}_\mathbf{T}^{n,3} + \mathbf{V}_\mathbf{T}^{n,2}) \cdot \frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} \\ &\quad + m(e_j^3)\frac{1}{8}(3\mathbf{V}_\mathbf{T}^{n,0} + \mathbf{V}_\mathbf{T}^{n,1} + 3\mathbf{V}_\mathbf{T}^{n,3} + \mathbf{V}_\mathbf{T}^{n,2}) \cdot \frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|}.\end{aligned} \tag{5.37}$$

The second term can be approximated utilizing relations (7.12), (7.13), (7.16), since $\Delta_\mathbf{x}\mathbf{x} = H\mathbf{N}$, as

$$\int_{\mathcal{V}_i} H\beta \,\mathrm{d}\mathcal{M} = \int_{\mathcal{V}_i} H\mathbf{N} \cdot \beta\mathbf{N} \,\mathrm{d}\mathcal{M} \approx \beta_i \mathbf{N}_i \cdot \sum_{j \in N_\square(\mathbf{x}_i)} \Big(m(e_j^1)\frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} + m(e_j^3)\frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|}\Big), \tag{5.38}$$

where the scalar β_i at the vertex $\mathbf{x}_i$ represents the normal evolution speed, and is defined as

$$\beta_i = \varepsilon_i H_i + \eta_i,$$

where

$$H_i = \mathbf{N}_i \cdot \sum_{j \in N_\square(\mathbf{x}_i)} \Big(m(e_j^1) \frac{\mathbf{N}_j^1}{|\mathbf{N}_j^1|} + m(e_j^3) \frac{\mathbf{N}_j^3}{|\mathbf{N}_j^3|} \Big) \Big/ m(\mathcal{V}_i).$$

The third term on the right-hand side in (5.36) represents the mean value of $H\beta$ along the surface and can be approximated as

$$\int_{\mathcal{V}_i} \frac{1}{A} \iint_{\mathcal{M}} H\beta \, \mathrm{d}\mathcal{M} \, \mathrm{d}\mathcal{M} \approx \frac{m(\mathcal{V}_i)}{A} \sum_{i=1}^{n} m(\mathcal{V}_i) H_i \beta_i, \tag{5.39}$$

where the area $A \approx \sum_{i=1}^{n} m(\mathcal{V}_i)$.

Regarding the fourth term on the right-hand side of (5.36), if we want in the discrete setting every finite volume $\mathcal{V}_i$ to have the same area, we would set $c = A$. The value of c is approximated as

$$c = A \approx \sum_{j=1}^{n} m(\mathcal{V}_j)$$

and the value of g is dependent on the number of quads forming the finite volume and it is approximated as $g \approx m(\mathcal{V}_i)/\Xi$, where $\Xi = \#N_\square(\mathbf{x}_i)/\sum_j^n \#N_\square(\mathbf{x}_j)$.Thus, the fourth term on the right-hand side of (5.36) can be approximated as

$$\begin{aligned} \int_{\mathcal{V}_i} \Big(\frac{c}{g} - 1 \Big) \omega_r \, \mathrm{d}\mathcal{M} &\approx \Big(\frac{A}{m(\mathcal{V}_i)/\Xi} - 1 \Big) \omega_r m(\mathcal{V}_i) \\ &= \Big(\frac{A(\#N_\square(\mathbf{x}_i))}{\sum_j^n \#N_\square(\mathbf{x}_j)} - m(\mathcal{V}_i) \Big) \omega_r. \end{aligned} \tag{5.40}$$

Putting the integral approximations into (5.36), we obtain a sparse linear system of n equations for the unknown values of the potential φ_i, $i = 1, \ldots, n$, at vertices $\mathbf{x}_i$: $A_\varphi \varphi^{t+\Delta t} = b_\varphi$, where $\varphi^{t+\Delta t} = (\varphi_1, \ldots, \varphi_n)^\top$. According to Corollary 5.1 the equation for the point $\mathbf{x}_1$ is replaced by the equation $\varphi_1 = 0$. The system is solved at every time step t.

5.4. Algorithm `Basic_Quad_Layout_evolution`**.** The Algorithm `Basic_Quad_Layout_evolution` summarizes the procedure described in Phase 3 to obtain from the Basic Quad Layout S a pure quad mesh approximation $M_\square$ of an input surface represented by $M_\triangle$. We propose two approaches to Phase 3 (Basic Quad Layout Evolution):

▷ `ALG_1` evolves each patch S_i of the surface Basic Quad Layout S separately, while fixing its boundary (imposing Dirichlet boundary conditions)

▷ `ALG_2` allows to evolve jointly also the boundary of each S_i, thus obtaining a better quad quality over the whole mesh as well as a better vertex distribution around the boundary of each S_i.

The strategy `ALG_1` turns out to be useful when specific parts of the shape are aimed to be edited while maintaining the prescribed partitioning boundaries given by the patch layout. This is the case, for example, of the B-Rep of objects in solid modelling.

For a more efficient implementation, `ALG_2` can be implemented as a post-process to `ALG_1` setting $\mathbf{x}^0 = M_\square$ obtained from `ALG_1`. This allows for both a parallel implementation and efficiency in the solutions of smaller-dimension linear systems. For all the experiments reported in the numerical section, we will refer to `ALG_2` as the latter, more efficient implementation.

Both strategies `ALG_1` and `ALG_2` stop the surface evolution when the following criterion is satisfied:

$$\|\mathbf{x}^{t+\Delta t} - \mathbf{x}^t\|_\infty \leqslant d_{th}$$

with d_{th} a given threshold.

Algorithm: `Basic_Quad_Layout_evolution`

Input: Basic Quad Layout $\{S_i\}_{i=1}^K$, partitioning $\{M_i\}_{i=1}^K$

Output: pure quad mesh $M_\square$

Parameters: time step τ,
redistribution speed parameters ω_r, ω_n

Preliminary Process:

· compute $d(\cdot)$ from $\{M_i\}_{i=1}^K$ using [40]

· enrich $d(\cdot)$ with a sign by a flooding algorithm.

`ALG_1`

for $i = 1, \ldots, K$ **do:**

· set $\mathbf{x}^0 = S_i$

· $t \leftarrow 0$

do: · compute $\beta\mathbf{N}$, $\mathbf{V}_\mathbf{T}^n$ using (5.26)–(5.27), (5.29)

· construct A_φ^t, b_φ^t using (5.36)–(5.40)

· solve $A_\varphi^t \varphi = b_\varphi^t$ for φ

· construct A_x^t, b_x^t using (5.26)–(5.27), (5.29)–(5.34)

· solve $A_x^t \mathbf{x}^{t+\Delta t} = b_x^t$ for $\mathbf{x}^{t+\Delta t}$, Dirichlet BC

· $t \leftarrow t + \Delta t$

while $\|\mathbf{x}^{t+\Delta t} - \mathbf{x}^t\|_\infty \leqslant d_{th}$

end for

ALG_2

for $i = 1, \ldots, K$ **do:**

- set $\mathbf{x}^0 = S$
- $t \leftarrow 0$

do:

- compute $\beta\mathbf{N}$, $\mathbf{V}_\mathbf{T}^n$ using (5.26)–(5.27), (5.29)
- construct A_φ^t, b_φ^t using (5.36)–(5.40)
- solve $A_\varphi^t \varphi = b_\varphi^t$ for φ
- construct A_x^t, b_x^t using (5.26)–(5.27), (5.29)–(5.34)
- solve $A_x^t \mathbf{x}^{t+\Delta t} = b_x^t$ for $\mathbf{x}^{t+\Delta t}$
- $t \leftarrow t + \Delta t$

while $\|\mathbf{x}^{t+\Delta t} - \mathbf{x}^t\|_\infty \leqslant d_{th}$

end for

6. Numerical experiments

We validated the proposed surface-patch quadrangulation algorithm on several input meshes. Data on the input triangular meshes $M_\triangle$ considered in the examples is reported in Table 1. From the second to the fifth column we report the number of vertices ($\#V_\triangle$), triangles ($\#T_\triangle$), number of patches K produced from the input mesh $M_\triangle$ by Phase 1, and the number of internal vertices that do not have six neighbours (extraordinary vertices $\#EV_\triangle$), which gives an insight into the input mesh structure. From now on we will refer to $V_\triangle$ for the vertices of the triangular mesh $M_\triangle$ and similarly $V_\square$ for the quad mesh vertices in $M_\square$.

	$\#V_\triangle$	$\#T_\triangle$	K	$\#EV_\triangle$
dolphin	7573	15142	11	2588
fertility	19994	40000	19	12092
hand	6607	13210	10	2827
horse	8078	16152	11	4712
plane	7470	14936	12	1076
pliers	5110	10216	4	1712
sole	11133	21946	1	8198
teddy	9548	19092	10	2822
teddy_2	1029	2056	10	633
wolf	4712	9420	17	2825

Table 1. Input data set $M_\triangle$.

Concerning the setting of the parameters involved in the quadrangulation algorithm, we proceed as follows. The edge length h in the initial grids of the Basic

Quad Layout S is set in the range $h \in [0.025, 0.04]$. However, the edge lengths at the end of the evolution step may slightly differ. In Phase 3, we set $c_1 = 0.03$, $c_2 = 0.002$ for the mean curvature flow weighting function ε in (5.11) and the initial time step $\tau = 1$. The tolerance d_{th} for the stopping criterion of Algorithm `Basic_Quad_Layout_evolution` is set to be $d_{th} = 5 \times 10^{-5}$. For the redistribution speeds ω_r and ω_n we chose the following strategy. In the inner `do` loop of Algorithm `Basic_Quad_Layout_evolution` we first set $\omega_r = \omega_n = 1$ to equally balance the area-oriented and angle-oriented redistributions and then we decreased $\tau = 0.5$, $\omega_r = 0.5$ and $\omega_n = 0.1$ when $\|\mathbf{x}^{t+\Delta t} - \mathbf{x}^t\|_\infty \leqslant 2 \times 10^{-4}$, thus obtaining a good compromise between the mesh quad areas and quad angles.

The code for Phase 1 and Phase 2 is written in MATLAB while the evolution in Phase 3 is implemented in C++ language. The experimental tests have been performed on Intel® Core™ i7-5820K 6-Core 3.30 GHz machine, with 64 GB/RAM and Radeon™ RX 460 graphics card in a Windows OS. The codes were executed without any additional machine support, e.g., parallelization, GPU support, register usage. The visualizations were created using ParaView and its VTK visualization toolkit.

6.1. Example 1: Mesh quadrangulation quality. In this experiment, we tested our algorithm on different meshes, investigating the quality of the results in terms of uniformity and closeness to the input shape $M_\triangle$. We impose $h = 0.025$ in Phase 2 for all examples in this experiment.

We measure the quad mesh quality by metrics on the final edge lengths h, the quad areas Q and the angles $\measuredangle$. In particular, we resort on the following metrics:

- ▷ $(\bar{h}, \sigma_h)$, mean edge length and standard deviation from $\bar{h}$,
- ▷ (Q, σ_Q), mean area of the quads and standard deviation from Q,
- ▷ $(\measuredangle, \sigma_\measuredangle)$, mean angle of the quad corners in degrees and standard deviation from $\measuredangle$.

Moreover, with respect to the reconstruction accuracy, we evaluate the following metrics on the distance function values:

- ▷ (d, σ_d), mean value of the distance function $d = n^{-1} \sum_{i=1}^{n} |d(\mathbf{x}_i)|$, $\mathbf{x}_i \in V_\square$, and the standard deviation,
- ▷ d_∞, the maximum distance $d_\infty := \|d(\mathbf{x})\|_\infty = \max_{i=1,\dots,n} |d(\mathbf{x}_i)|$, $\mathbf{x}_i \in V_\square$.

In Figures 8–9 we report some examples of the quadrilateral meshes obtained by running the Algorithm `Surface-Patch Quadrangulation` where both the proposals `ALG_1` and `ALG_2` have been applied in Phase 3. For each mesh in Figures 8–9 we visualize in the first column the input Basic Quad Layout S together with an outline of the input shape, where each patch S_i of S is coloured differently; in the second and the third column, we report the outputs of `ALG_1` and `ALG_2`, respectively, coloured

Figure 8. Example 1: Reconstruction results of different meshes. From left to right: input Basic Quad Layout S with the outline of $M_{\triangle}$ in transparent, result of `ALG_1` and result of `ALG_2`.

in false colours in the range [blue,red] according to the quad areas of the mesh. As expected, in the third column of Figures 8–9, which illustrates the output of `ALG_2`, we can appreciate the uniform colour distribution that was obtained thanks to the boundary moving strategy. Moreover, even when `ALG_2` is applied, the original

Figure 9. Example 1: Reconstruction results of different meshes. From left to right: input Basic Quad Layout S with the outline of $M_\triangle$ in transparent, result of `ALG_1` and result of `ALG_2`.

patch layout is well-preserved. The details of the resulting meshes $M_\square$ illustrated in Figure 8–9 are listed in Table 2. The number of extraordinary vertices ($\#EV_\square$) is much lower than $\#EV_\triangle$ of the input mesh $M_\triangle$ in Table 1, indicating that the results $M_\square$ are valence semi-regular meshes. Moreover, even for different values of the edge length h used, thanks to the construction of S in Phase 2, the number of extraordinary vertices remains unchanged.

	$\#V_\square$	$\#Q_\square$	$\#EV_\square$
dolphin	7482	7480	60
fertility	6768	6774	113
hand	7200	7198	52
horse	5164	5162	58
plane	8057	8055	64
pliers	2649	2647	20
sole	21442	21070	4
teddy	8610	8608	56
teddy_2	6018	6016	52
wolf	14348	14346	92

Table 2. Example 1: Output data set $M_\square$ for meshes illustrated in Figures 8 and 9.

In addition to the qualitative evaluation illustrated in Figures 8–9, we report in Table 3 the quality measures for both ALG_1 and ALG_2 marked by upper indices for each mesh name in the first column. Both algorithms produce well-shaped quad mesh elements in terms of angles (almost 90 degrees in the sixth column $\measuredangle$) and area density (small value of σ_Q). ALG_2 allows for a more uniform quad area distribution (σ_Q is smaller for ALG_2 than for ALG_1) and a more accurate reconstruction (σ_d is smaller for ALG_2). Let us notice that although we did not expect h to be preserved, the mean edge length $\bar{h}$ is relatively close to the chosen h.

The last two columns of Table 3 report the timing for the evolution Phase 3 in terms of the number of iterations ($\#its$) needed for convergence to the given threshold d_{th} and time per one iteration in milliseconds. In particular, the $\#its$ column for ALG_1 reports the medium number of iterations per patch while for ALG_2 $\#its$ reports the number of iterations needed as the postprocess to $M_\square$ of ALG_1. We notice that the number of iterations naturally depends on the Basic Quad Layout and on the distance to be reached by the evolution, see e.g. *pliers* and plane, w.r.t. shorter distances as in teddy or dolphin.

6.2. Example 2: Effect of the tangential redistribution terms. To evaluate the benefit of the tangential redistributions during the evolution in Phase 3, we first evolved the Basic Quad Layout S of the teddy mesh using ALG_2 balancing the quad areas and angle distortions. In Figure 10 (first row) the evolution of one patch S_i of S in four selected time steps is reported. Then we re-ran the algorithm first without the angle contribution $\mathbf{V}_\mathbf{T}^n$ term and then without the area contribution $\mathbf{V}_\mathbf{T}^r$. The comparisons related to the resulting patch S_i w.r.t. the balanced first evolution are reported in the second row in Figure 10. On the left (result without the $\mathbf{V}_\mathbf{T}^n$ term), the angle-distortion values $\delta(\measuredangle)$ in false colours are over-imposed on the quad mesh.

The angle-distortion $\delta(\measuredangle)$ at a vertex $\mathbf{x}_i$ is computed as

$$\delta(\measuredangle)|_{\mathbf{x}_i} = \frac{\sum_{j \in N_\square(\mathbf{x}_i)} |\gamma_j - \frac{1}{2}\pi|}{\#N_\square(\mathbf{x}_i)},$$

where γ_j represents the angle size at $\mathbf{x}_i$ in the neighbouring quad Q_j.

On the right (result without the $\mathbf{V}_\mathbf{T}^r$ term), the quads are coloured according to their areas $m(Q)$ using false colours. In the last row of Figure 10, we report the histograms corresponding to the value distributions shown in the second row of Figure 10. The blue bins represent the value distribution without the relative tangential term, while the red bins correspond to the resulting values using both tangential contributions.

We can conclude that both tangential terms are important for the quality of the resulting mesh elements, while the leading contribution is represented by the area redistribution term $\mathbf{V}_\mathbf{T}^r$.

	(h, σ_h)		(Q, σ_Q)		$(\measuredangle, \sigma_\measuredangle)$		(d, σ_d)		d_∞	$\#its$	$1\,its$
	$\times 10^{-3}$		$\times 10^{-5}$		deg		$\times 10^{-6}$		$\times 10^{-5}$		msec
dolphin[1]	20.11	5.59	35.07	3.26	89.9	21.9	−0.55	7.43	6.40	44	11
dolphin[2]	20.02	5.56	35.06	0.71	89.9	19.9	−0.08	1.06	0.33	61	135
fertility[1]	16.42	4.31	22.22	2.70	89.7	26.7	0.00	0.47	2.13	53	7
fertility[2]	15.91	4.15	22.17	1.09	89.8	20.9	0.00	0.00	0.01	60	130
hand[1]	22.15	5.05	43.41	3.95	89.9	21.9	−0.61	9.49	15.30	116	9
hand[2]	22.04	5.21	43.38	0.71	89.9	19.4	−0.01	0.78	1.06	60	121
horse[1]	23.63	6.85	48.28	6.18	89.8	21.5	−6.89	22.25	9.94	59	6
horse[2]	23.63	6.85	48.22	1.19	89.8	19.6	−0.11	1.76	2.43	63	82
plane[1]	18.77	5.88	29.70	1.61	89.9	22.2	0.00	0.04	0.50	104	15
plane[2]	18.53	5.87	29.70	0.83	89.9	18.6	0.00	0.04	0.18	50	154
pliers[1]	30.12	16.20	61.80	2.75	89.9	19.2	0.01	0.02	0.24	525	9
pliers[2]	30.17	16.16	61.80	1.70	89.9	17.4	0.01	0.01	0.20	34	48
teddy[1]	22.18	6.80	41.75	2.80	89.9	22.5	−2.25	17.84	2.20	47	13
teddy[2]	22.04	6.78	41.78	0.77	89.9	20.0	−0.20	0.97	1.67	48	164
wolf[1]	15.88	5.16	20.70	1.60	89.8	22.4	−0.06	7.36	36.40	86	11
wolf[2]	15.54	4.98	20.70	0.63	89.8	18.0	−0.04	0.62	6.40	147	248

Table 3. Example 1: Reconstruction quality metrics for the meshes reported in Figures 8 and 9.

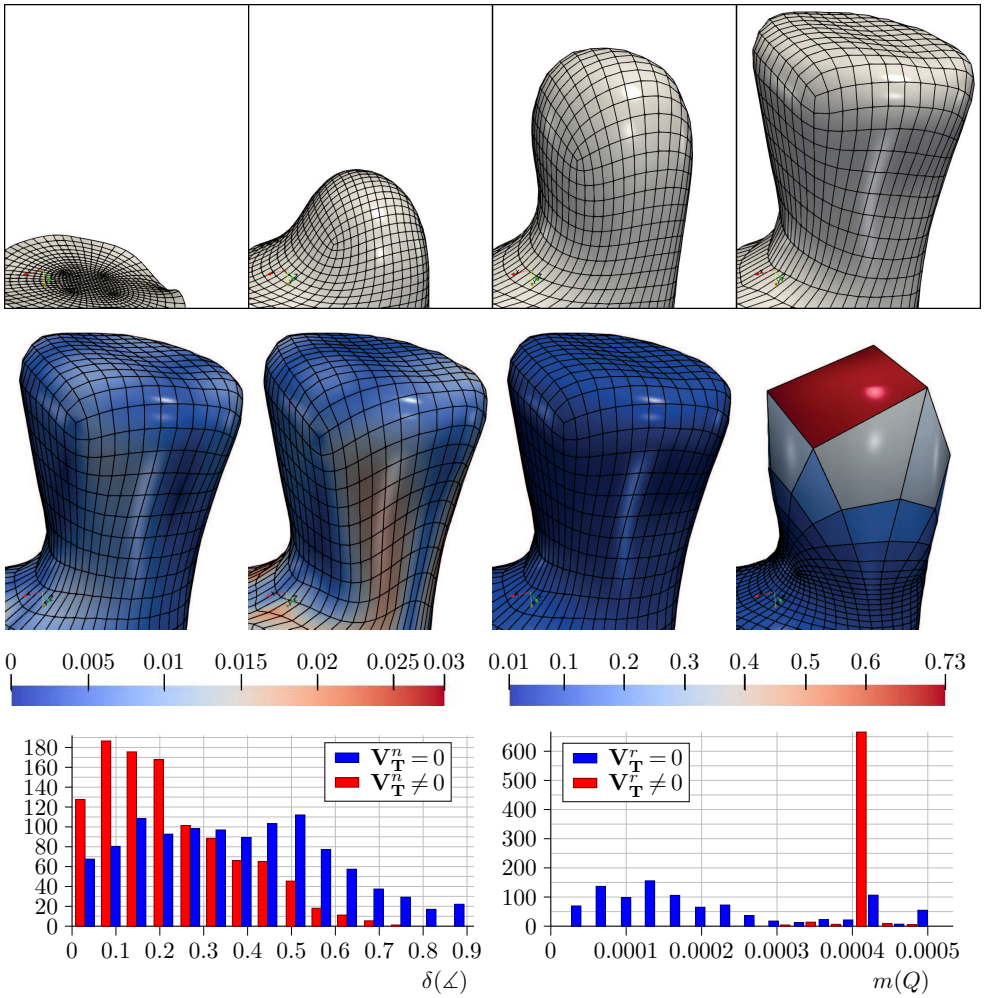


Figure 10. Example 2: Demonstration of the influence of the tangential redistribution terms. First row: Selected time steps of patch evolution for `teddy` mesh. Second row (left): Comparison of the resulting mesh part with (left) and without (right) the **angle** redistribution term. Second row (right): Comparison of the resulting mesh part with (left) and without (right) the **area** redistribution term. Third row: corresponding histogram value distributions relative to the results in the second row.

6.3. Example 3: Robustness to the mesh resolution. In this experiment we investigated the robustness of the proposed algorithm `ALG_2` to the changes on the input ($M_\triangle$) and output ($M_\square$) mesh resolutions. To this aim we considered two input meshes $M_\triangle$ of different resolutions: `teddy` (high density) and `teddy_2` (low density), see Figure 11 top left and bottom left, respectively. To produce `teddy_2`, we have decreased the input $M_\triangle$ mesh resolution of `teddy` down to 10%, see `teddy_2` in Table 1.

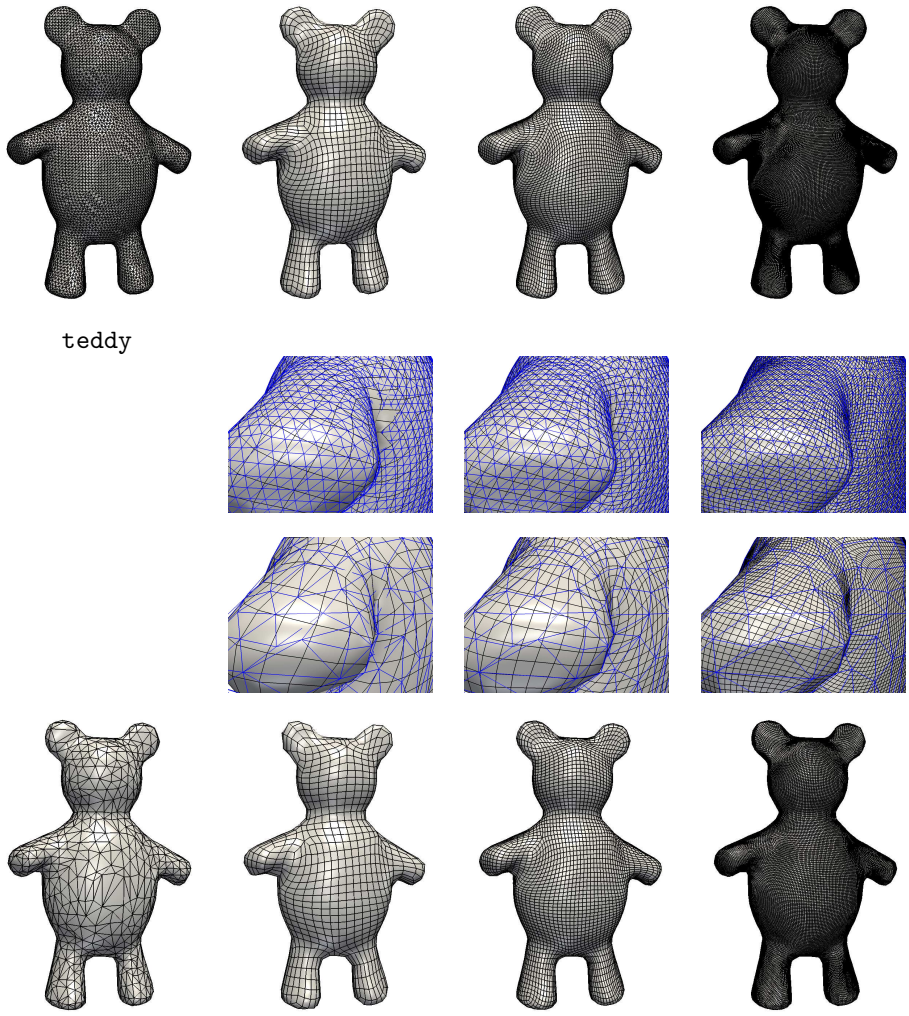


Figure 11. Example 3: Quad mesh results under different resolutions for `teddy` mesh (top left) and `teddy_2` mesh (bottom left). From left to right the results $M_\square$ for chosen edge lengths $h = \{0.04, 0.02, 0.01\}$, together with details of the reconstructed meshes with original triangulations over-imposed in blue (second and third row).

We produced the Basic Quad Layout S for both the `teddy` meshes using three different resolutions controlled by decreasing h in the range $\{0.04, 0.02, 0.01\}$ and we evolved the associated layouts. The corresponding results $M_\square$ of `ALG_2` are reported in Figure 11 in the first and last row. Each resolution result in Figure 11 is accompanied with a detail of the arm together with the original triangulation coloured

in blue. From the visual inspection of the reported results we can state that our algorithm is robust to the input-output mesh resolution.

In order to evaluate the accuracy of the reconstructions, we measured the Hausdorff distance, implemented in the software Metro [5], which is defined between two meshes X and Y as

$$d_H(X,Y) = \max\Big\{ \sup_{x\in X}\Big(\inf_{y\in Y} \operatorname{dist}(x,y)\Big),\ \sup_{y\in Y}\Big(\inf_{x\in X} \operatorname{dist}(x,y)\Big)\Big\}.$$

The Hausdorff distances between $M_\triangle$ and $M_\square$ for the meshes illustrated in Figure 11 are reported in Table 4. In the last column (`teddy`) and the last row (`teddy_2`) of Table 4 we report the distances between the input triangulation $M_\triangle$ and the corresponding quad mesh results $M_\square$ for the different edge length parameter values h. Decreasing the edge length h, the reconstructed shape $M_\square$ improves, as confirmed by the decreasing Hausdorff distances.

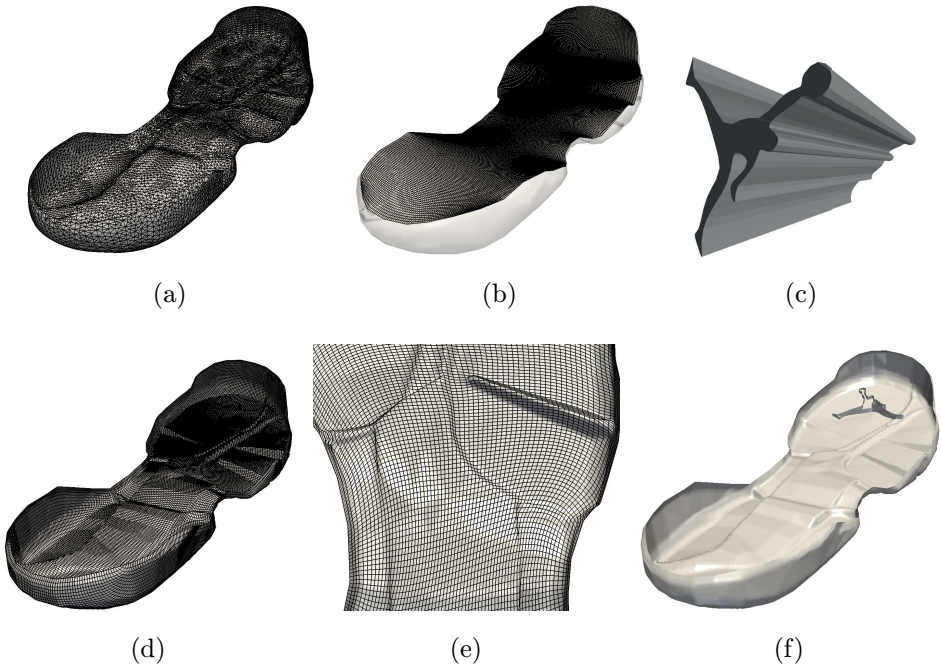


Figure 12. Example 4: quad mesh generation from 3D scanned objects: (a) input triangulation $M_\triangle$; (b) the Basic Quad Layout S; (c) "logo" NURBS surface; (d) $M_\square$; (e) a detail of $M_\square$; (f) Boolean operation between the surfaces.

In the central part of Table 4, at position $(1,1)$ the reader can find the distance between the two input triangulations, while the rest of the first row lists the Hausdorff distances between the `teddy` input triangulation $M_\triangle$ and the resulting meshes $M_\square$

for teddy_2 and vice versa for the rest of the first column. Also in this case, we notice that for smaller values of h, the given shape is better approximated, thus, approaching the same distance as between the input triangulations $M_\triangle$. The rest of the central part of Table 4 reports for different values of h the distances between the reconstructed $M_\square$ for teddy and the resulting $M_\square$ for teddy_2.

			teddy_2			teddy
		$M_\triangle$	$M_\square$	$M_\square$	$M_\square$	
$d_H(X,Y) \times 10^{-3}$			$h = 0.04$	$h = 0.02$	$h = 0.01$	$d_H(M_\triangle, M_\square)$
teddy	$M_\triangle$	14.50	28.31	16.95	14.50	0.00
	$M_\square$, $h = 0.04$	17.27	28.32	17.38	18.14	22.60
	$M_\square$, $h = 0.02$	12.99	26.02	15.48	13.38	12.78
	$M_\square$, $h = 0.01$	13.68	28.57	16.54	13.68	3.01
teddy_2	$d_H(M_\triangle, M_\square)$	0.00	22.68	13.58	5.80	

Table 4. Example 3: Hausdorff distances between the results reported in Figure 11.

6.4. Example 4: Sole. Finally we illustrate a simple example of CAD application where the Boolean operation between two NURBS surfaces (the "logo" and the sole) can be realized by first remeshing the 3D scanned object (representing the sole) into a quad regular grid, thus providing a NURBS representation.

The 3D scanned object, representing a sole of a shoe, given as a triangular mesh $M_\triangle$, is shown in Figure 12(a) (see Table 1 for detailed data). The 1-patch Basic Quad Layout S constructed from the boundary of $M_\triangle$ (see Figure 12(b) where the input shape is over-imposed in semi-transparency) is then evolved by the proposed quad mesh generation algorithm. The resulting mesh $M_\square$ is illustrated in Figure 12(d) together with a zoomed detail of the object in Figure 12(e), and the corresponding data is reported in Table 2. The quad mesh sole and the "logo" are then combined together by Boolean intersection obtaining the surface in Figure 12(f).

7. Conclusion

The generation of pure quad meshes built on a consistent object patch-layout is of paramount importance on a variety of application domains ranging from Finite Element Analysis to solid modelling and computer graphics. In general, quad meshes hold the potential to simplify and improve efficiency of many algorithms in surface processing. In the context of FEM, the geometry is often available in the form of quad patches, each preferably represented by quad elements as uniform as possible with respect to areas and right angles. Preliminarily to many editing processes, CAD

applications require to convert a 3D scanned object into a minimal set of salient patches easily treatable in terms of NURBS or of T-spline surfaces.

Towards these targets, we introduced a framework which, first, constructs a Basic Quad Layout underlying a 0-genus partitioning of an input discrete manifold, and then evolves it following a Lagrangian-type evolution model to finally generate a patch-based pure quad mesh which accurately approximates the original shape. The evolution model is robust, however, in order to globally preserve a uniform quad distribution it only considers 0-genus patches with one boundary. Extensions to cylindrical 0-genus patches, produced by a possibly more sophisticated partitioning method, are left for further investigation. Our algorithm guarantees various aspects of the resulting meshes, such as the low number of singularities and patches, and the quality of geometry approximation. Concerning the quad element quality it provides a good compromise between uniform area and right angle distribution. However, the edge length fixed on the patch boundaries in Phase 2 is not eventually preserved during the evolution. A fully satisfactory, future, quad mesh generation method could include the edge length control into the evolution model. Finally, we observe that our framework is capable of extracting meaningful Basic Quad Layouts from complex shapes which represent precious shape descriptors. This is an interesting topic for future investigations.

Appendix

(A) P r o o f of Lemma 5.1. The local density g of the surface evolving by (5.1) can be controlled by the tangential movement. In our case by a right choice of parameters α and λ in (5.2). The relation of how g changes in time depending on α and λ (and prescribed β) is derived in the sequel.

Starting with the following relation between g_t, $\mathbf{x}_u$ and $\mathbf{x}_v$:

$$g_t = |\mathbf{x}_u \times \mathbf{x}_v|_t = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot (\mathbf{x}_u \times \mathbf{x}_v)_t, \tag{7.1}$$

the term $(\mathbf{x}_u \times \mathbf{x}_v)_t$ can be expanded as

$$\begin{aligned}(\mathbf{x}_u \times \mathbf{x}_v)_t &= (\mathbf{x}_u)_t \times \mathbf{x}_v + \mathbf{x}_u \times (\mathbf{x}_v)_t \\ &= (\mathbf{x}_t)_u \times \mathbf{x}_v + \mathbf{x}_u \times (\mathbf{x}_t)_v \\ &= (\beta \boldsymbol{N} + \alpha \mathbf{x}_u + \lambda \mathbf{x}_v)_u \times \mathbf{x}_v + \mathbf{x}_u \times (\beta \boldsymbol{N} + \alpha \mathbf{x}_u + \lambda \mathbf{x}_v)_v \\ &= (\beta \boldsymbol{N})_u \times \mathbf{x}_v + (\alpha \mathbf{x}_u + \lambda \mathbf{x}_v)_u \times \mathbf{x}_v \\ &\qquad + \mathbf{x}_u \times (\beta \boldsymbol{N})_v + \mathbf{x}_u \times (\alpha \mathbf{x}_u + \lambda \mathbf{x}_v)_v.\end{aligned} \tag{7.2}$$

For clarity reasons, let us further focus on (7.2) split into two parts, by collecting the terms containing $\beta \boldsymbol{N}$ and the ones containing $\alpha \mathbf{x}_u + \lambda \mathbf{x}_v$.

Plugging the first part of (7.2) into (7.1), we obtain

$$\begin{aligned}
&\frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot ((\beta \mathbf{N})_u \times \mathbf{x}_v + \mathbf{x}_u \times (\beta \mathbf{N})_v) \\
&\qquad = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot (\beta_u \mathbf{N} \times \mathbf{x}_v + \beta \mathbf{N}_u \times \mathbf{x}_v + \mathbf{x}_u \times \beta_v \mathbf{N} + \mathbf{x}_u \times \beta \mathbf{N}_v) \\
&\qquad = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot (\beta \mathbf{N}_u \times \mathbf{x}_v + \mathbf{x}_u \times \beta \mathbf{N}_v) \\
&\qquad = \beta \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot (\mathbf{N}_u \times \mathbf{x}_v + \mathbf{x}_u \times \mathbf{N}_v) = -g \; H \beta,
\end{aligned}$$

where the formula from [13] (Lemma 13.38, page 411) has been used for the mean curvature.

Similarly, plugging the second, tangential, part of (7.2) into (7.1) we get

$$\begin{aligned}
&\frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot ((\alpha \mathbf{x}_u + \lambda \mathbf{x}_v)_u \times \mathbf{x}_v + \mathbf{x}_u \times (\alpha \mathbf{x}_u + \lambda \mathbf{x}_v)_v) \\
&\quad = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot ((\alpha_u \mathbf{x}_u + \alpha \mathbf{x}_{uu} + \lambda_u \mathbf{x}_v + \lambda \mathbf{x}_{vu}) \times \mathbf{x}_v \\
&\qquad\qquad + \mathbf{x}_u \times (\alpha_v \mathbf{x}_u + \alpha \mathbf{x}_{uv} + \lambda_v \mathbf{x}_v + \lambda \mathbf{x}_{vv})) \\
&\quad = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot (\alpha_u \mathbf{x}_u \times \mathbf{x}_v + \alpha \mathbf{x}_{uu} \times \mathbf{x}_v + \lambda_u \mathbf{x}_v \times \mathbf{x}_v + \lambda \mathbf{x}_{vu} \times \mathbf{x}_v \\
&\qquad\qquad + \alpha_v \mathbf{x}_u \times \mathbf{x}_u + \alpha \mathbf{x}_u \times \mathbf{x}_{uv} + \lambda_v \mathbf{x}_u \times \mathbf{x}_v + \lambda \mathbf{x}_u \times \mathbf{x}_{vv}) \\
&\quad = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \cdot (\alpha_u \mathbf{x}_u \times \mathbf{x}_v + \alpha \mathbf{x}_{uu} \times \mathbf{x}_v + \lambda \mathbf{x}_{vu} \times \mathbf{x}_v \\
&\qquad\qquad + \alpha \mathbf{x}_u \times \mathbf{x}_{uv} + \lambda_v \mathbf{x}_u \times \mathbf{x}_v + \lambda \mathbf{x}_u \times \mathbf{x}_{vv}) \\
&\quad = \left(\alpha_u |\mathbf{x}_u \times \mathbf{x}_v| + \alpha \frac{(\mathbf{x}_u \times \mathbf{x}_v) \cdot (\mathbf{x}_{uu} \times \mathbf{x}_v + \mathbf{x}_u \times \mathbf{x}_{uv})}{|\mathbf{x}_u \times \mathbf{x}_v|} \right. \\
&\qquad\qquad \left. + \lambda_v |\mathbf{x}_u \times \mathbf{x}_v| + \lambda \frac{(\mathbf{x}_u \times \mathbf{x}_v) \cdot (\mathbf{x}_u \times \mathbf{x}_{vv} + \mathbf{x}_{vu} \times \mathbf{x}_v)}{|\mathbf{x}_u \times \mathbf{x}_v|} \right) \\
&\quad = \left(\alpha_u |\mathbf{x}_u \times \mathbf{x}_v| + \alpha \frac{(\mathbf{x}_u \times \mathbf{x}_v) \cdot (\mathbf{x}_u \times \mathbf{x}_v)_u}{|\mathbf{x}_u \times \mathbf{x}_v|} \right. \\
&\qquad\qquad \left. + \lambda_v |\mathbf{x}_u \times \mathbf{x}_v| + \lambda \frac{(\mathbf{x}_u \times \mathbf{x}_v) \cdot (\mathbf{x}_u \times \mathbf{x}_v)_v}{|\mathbf{x}_u \times \mathbf{x}_v|} \right) \\
&\quad = (\alpha_u |\mathbf{x}_u \times \mathbf{x}_v| + \alpha |\mathbf{x}_u \times \mathbf{x}_v|_u + \lambda_v |\mathbf{x}_u \times \mathbf{x}_v| + \lambda |\mathbf{x}_u \times \mathbf{x}_v|_v) \\
&\quad = \frac{|\mathbf{x}_u \times \mathbf{x}_v|}{|\mathbf{x}_u \times \mathbf{x}_v|} ((\alpha |\mathbf{x}_u \times \mathbf{x}_v|)_u + (\lambda |\mathbf{x}_u \times \mathbf{x}_v|)_v) = g \nabla_{\mathbf{x}} \cdot \mathbf{V}_{\mathbf{T}},
\end{aligned}$$

where the formula for the divergence of a vector field (5.8) has been used in the last step. Combining the preceding two parts together, we obtain (5.17), which completes the proof. □

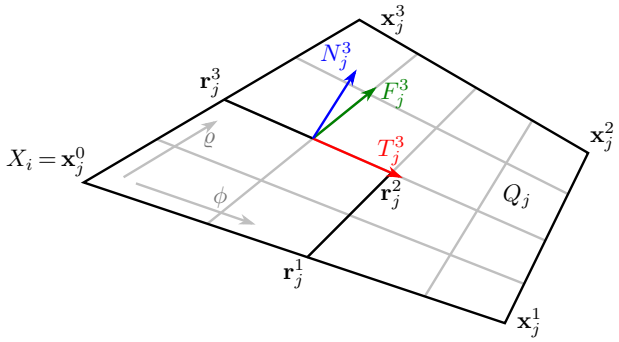


Figure 13. Illustration of the bilinear interpolation at one quad Q_j, $j \in N_\square(\mathbf{x}_i)$ used in the approximation of normal derivatives.

(B) P r o o f of Theorem 5.1. The proof is based on expanding the left-hand side of (5.18) to

$$\left(\frac{g}{A}\right)_t = \frac{g_t A - g A_t}{A^2}, \tag{7.3}$$

where g_t is given by (5.17), and for A_t we obtain

$$\begin{aligned}
A_t &= \left(\int_0^1 \int_0^1 g \,\mathrm{d}u \,\mathrm{d}v\right)_t = \int_0^1 \int_0^1 g_t \,\mathrm{d}u \,\mathrm{d}v \\
&= -\int_0^1 \int_0^1 g \;H\beta \,\mathrm{d}u \,\mathrm{d}v + \int_0^1 \int_0^1 g \;\nabla_{\mathbf{x}} \cdot \mathbf{V_T} \,\mathrm{d}u \,\mathrm{d}v \\
&= -\iint_{\mathcal{M}} H\beta \,\mathrm{d}\mathcal{M} + \iint_{\mathcal{M}} \nabla_{\mathbf{x}} \cdot \mathbf{V_T} \,\mathrm{d}\mathcal{M} \\
&= -\iint_{\mathcal{M}} H\beta \,\mathrm{d}\mathcal{M} + \oint_{\partial\mathcal{M}} \mathbf{V_T} \cdot \mathbf{n} \,\mathrm{d}S \\
&= -\iint_{\mathcal{M}} H\beta \,\mathrm{d}\mathcal{M} + 0,
\end{aligned}$$

where the Gauss theorem was used. Since our surface $\mathcal{M}$ is either a closed surface or a surface with a boundary on which the zero Neumann boundary condition (5.20) is prescribed, the boundary integral in the above formulation is equal to zero.

Thus, collecting the obtained expressions, we can write

$$\left(\frac{g}{A}\right)_t = \frac{(-gH\beta + g\nabla_{\mathbf{x}} \cdot \mathbf{V_T})A + g\iint_{\mathcal{M}} H\beta \,\mathrm{d}\mathcal{M}}{A^2}. \tag{7.4}$$

Then, plugging (7.4) into equation (5.18), we get

$$\frac{(-g\;H\beta + g\;\nabla_{\mathbf{x}} \cdot \mathbf{V_T})A + g\iint_{\mathcal{M}} H\beta \,\mathrm{d}\mathcal{M}}{A^2} = \left(\frac{c}{A} - \frac{g}{A}\right)\omega_r. \tag{7.5}$$

Using (5.13) for $\mathbf{V_T}$ in (7.5) and after simple rearrangement we obtain (5.19). □

(C) Numerical approximation of the normal derivatives. In the following we describe the bilinear interpolation used to approximate the normal derivatives $\partial f/\partial \mathbf{n}_j^1$, $\partial f/\partial \mathbf{n}_j^3$ in (5.24), (5.31), (5.32), (5.37) of a generic scalar function f. An illustration of the following definitions is presented in Figure 13.

Let us recall Green's theorem applied to $\int_{\mathcal{V}_i} \Delta_{\mathbf{x}} f \,\mathrm{d}S$ that gives rise to the normal derivatives

$$\int_{\partial \mathcal{V}_i} \nabla_{\mathbf{x}} f \cdot \mathbf{n} \,\mathrm{d}s = \sum_{j \in N_\square(\mathbf{x}_i)} \int_{e_j^1} \frac{\partial f}{\partial \mathbf{n}_j^1} \,\mathrm{d}s + \int_{e_j^3} \frac{\partial f}{\partial \mathbf{n}_j^3} \,\mathrm{d}s. \tag{7.6}$$

The points on Q_j, $j \in N_\square(\mathbf{x}_i)$ can be expressed by

$$\mathbf{x}(\phi,\varrho) = (1-\phi)(1-\varrho)\mathbf{x}_j^0 + \phi(1-\varrho)\mathbf{x}_j^1 + (1-\phi)\varrho\mathbf{x}_j^3 + \phi\varrho\mathbf{x}_j^2,\ \phi \in [0,1],\ \varrho \in [0,1], \tag{7.7}$$

and similarly the values of f by

$$f(\phi,\varrho) = (1-\phi)(1-\varrho)f_j^0 + \phi(1-\varrho)f_j^1 + (1-\phi)\varrho f_j^3 + \phi\varrho f_j^2,\ \phi \in [0,1],\ \varrho \in [0,1]. \tag{7.8}$$

Derivatives of $\mathbf{x}(\phi,\varrho)$ are

$$\begin{aligned} \frac{\partial \mathbf{x}}{\partial \varrho}(\phi,\varrho) &= -(1-\phi)\mathbf{x}_j^0 - \phi\mathbf{x}_j^1 + (1-\phi)\mathbf{x}_j^3 + \phi\mathbf{x}_j^2, \\ \frac{\partial \mathbf{x}}{\partial \phi}(\phi,\varrho) &= -(1-\varrho)\mathbf{x}_j^0 + (1-\varrho)\mathbf{x}_j^1 - \varrho\mathbf{x}_j^3 + \varrho\mathbf{x}_j^2. \end{aligned} \tag{7.9}$$

The non-normalized tangents to the edges e_j^1 and e_j^3 are defined as

$$\begin{aligned} \boldsymbol{T}_j^1 &= \frac{\partial \mathbf{x}}{\partial \varrho}\Big(\frac{1}{2},\frac{1}{4}\Big) = -\frac{1}{2}\mathbf{x}_j^0 - \frac{1}{2}\mathbf{x}_j^1 + \frac{1}{2}\mathbf{x}_j^3 + \frac{1}{2}\mathbf{x}_j^2, \\ \boldsymbol{T}_j^3 &= \frac{\partial \mathbf{x}}{\partial \phi}\Big(\frac{1}{4},\frac{1}{2}\Big) = -\frac{1}{2}\mathbf{x}_j^0 + \frac{1}{2}\mathbf{x}_j^1 - \frac{1}{2}\mathbf{x}_j^3 + \frac{1}{2}\mathbf{x}_j^2. \end{aligned} \tag{7.10}$$

The derivatives of $\mathbf{x}$ define also the following two vectors:

$$\begin{aligned} \boldsymbol{F}_j^1 &= \frac{\partial \mathbf{x}}{\partial \phi}\Big(\frac{1}{2},\frac{1}{4}\Big) = -\frac{3}{4}\mathbf{x}_j^0 + 3/4\mathbf{x}_j^1 - 1/4\mathbf{x}_j^3 + 1/4\mathbf{x}_j^2, \\ \boldsymbol{F}_j^3 &= \frac{\partial \mathbf{x}}{\partial \varrho}\Big(\frac{1}{4},\frac{1}{2}\Big) = -\frac{3}{4}\mathbf{x}_j^0 - \frac{1}{4}\mathbf{x}_j^1 + \frac{3}{4}\mathbf{x}_j^3 + \frac{1}{4}\mathbf{x}_j^2. \end{aligned} \tag{7.11}$$

Using the vectors defined above, we can compute non-normalized normals to the edge e_j^1 using the Gram-Schmidt orthogonalization process as

$$\begin{aligned} \boldsymbol{N}_j^1 &= \boldsymbol{F}_j^1 - \frac{\boldsymbol{F}_j^1 \cdot \boldsymbol{T}_j^1}{\boldsymbol{T}_j^1 \cdot \boldsymbol{T}_j^1}\boldsymbol{T}_j^1 \\ &= -\frac{3}{4}\mathbf{x}_j^0 + \frac{3}{4}\mathbf{x}_j^1 - \frac{1}{4}\mathbf{x}_j^3 + \frac{1}{4}\mathbf{x}_j^2 - a_j^1\Big(-\frac{1}{2}\mathbf{x}_j^0 - \frac{1}{2}\mathbf{x}_j^1 + \frac{1}{2}\mathbf{x}_j^3 + \frac{1}{2}\mathbf{x}_j^2\Big), \end{aligned} \tag{7.12}$$

where $a_j^1 = (\boldsymbol{F}_j^1 \cdot \boldsymbol{T}_j^1)/(\boldsymbol{T}_j^1 \cdot \boldsymbol{T}_j^1)$. Similarly for the edge e_j^3 as

$$\begin{aligned}(7.13)\qquad \boldsymbol{N}_j^3 &= \boldsymbol{F}_j^3 - \frac{\boldsymbol{F}_j^3 \cdot \boldsymbol{T}_j^3}{\boldsymbol{T}_j^3 \cdot \boldsymbol{T}_j^3}\boldsymbol{T}_j^3 \\ &= -\frac{3}{4}\mathbf{x}_j^0 - \frac{1}{4}\mathbf{x}_j^1 + \frac{3}{4}\mathbf{x}_j^3 + \frac{1}{4}\mathbf{x}_j^2 - a_j^3\Big(-\frac{1}{2}\mathbf{x}_j^0 + \frac{1}{2}\mathbf{x}_j^1 - \frac{1}{2}\mathbf{x}_j^3 + \frac{1}{2}\mathbf{x}_j^2\Big),\end{aligned}$$

where $a_j^3 = (\boldsymbol{F}_j^3 \cdot \boldsymbol{T}_j^3)/(\boldsymbol{T}_j^3 \cdot \boldsymbol{T}_j^3)$.

At last, the derivative of f in the normal direction at the edge e_j^1 is approximated as

$$(7.14)\qquad \frac{\partial f}{\partial \mathbf{n}_j^1} \approx \Big(-\frac{3}{4}f_j^0 + \frac{3}{4}f_j^1 - \frac{1}{4}f_j^3 + \frac{1}{4}f_j^2 - \frac{a_j^1}{2}(-f_j^0 - f_j^1 + f_j^3 + f_j^2)\Big)/|\boldsymbol{N}_j^1|,$$

and similarly for $\partial f/\partial \mathbf{n}_j^3$. Now we can write the approximation of the Laplace-Beltrami operator applied on a generic function f and integrated over the finite volume $\mathcal{V}_i$ as

$$\begin{aligned}(7.15)\qquad &\int_{\partial\mathcal{V}_i} \nabla_s f \cdot \mathbf{n}\,\mathrm{d}s \\ &\approx \sum_{j\in N_\square(\mathbf{x}_i)} \frac{m(e_j^1)}{|\mathbf{N}_j^1|}\Big[\frac{1}{4}(-3f_j^0 + 3f_j^1 - f_j^3 + f_j^2) - \frac{a_j^1}{2}(-f_j^0 - f_j^1 + f_j^3 + f_j^2)\Big] \\ &\quad + \frac{m(e_j^3)}{|\mathbf{N}_j^3|}\Big[\frac{1}{4}(-3f_j^0 - f_j^1 + 3f_j^3 + f_j^2) - \frac{a_j^3}{2}(-f_j^0 + f_j^1 - f_j^3 + f_j^2)\Big],\end{aligned}$$

while the mean curvature integral, as used e.g. in (5.26), is approximated as

$$\begin{aligned}(7.16)\qquad &\int_{\partial\mathcal{V}_i} \nabla_s \mathbf{x} \cdot \mathbf{n}\,\mathrm{d}s \\ &\approx \sum_{j\in N_\square(\mathbf{x}_i)} \frac{m(e_j^1)}{|\mathbf{N}_j^1|}\Big[\frac{1}{4}(-3\mathbf{x}_j^0 + 3\mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2) - \frac{a_j^1}{2}(-\mathbf{x}_j^0 - \mathbf{x}_j^1 + \mathbf{x}_j^3 + \mathbf{x}_j^2)\Big] \\ &\quad + \frac{m(e_j^3)}{|\mathbf{N}_j^3|}\Big[\frac{1}{4}(-3\mathbf{x}_j^0 - \mathbf{x}_j^1 + 3\mathbf{x}_j^3 + \mathbf{x}_j^2) - \frac{a_j^3}{2}(-\mathbf{x}_j^0 + \mathbf{x}_j^1 - \mathbf{x}_j^3 + \mathbf{x}_j^2)\Big].\end{aligned}$$

References

[1] *J. W. Barrett, H. Garcke, R. Nürnberg*: On the parametric finite element approximation of evolving hypersurfaces in $\mathbb{R}^3$. J. Comput. Phys. *227* (2008), 4281–4307.

[2] *D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, D. Zorin*: Quad-Mesh generation and processing: A survey. Comput. Graph. Forum *32* (2013), 51–76.

[3] *D. Bommes, H. Zimmer, L. Kobbelt*: Mixed-integer quadrangulation. ACM Trans. Graph. *28* (2009), Article ID 77, 10 pages.

[4] *M. Campen*: Partitioning surfaces into quadrilateral patches: A survey. Comput. Graph. Forum *36* (2017), 567–588.

[5] *P. Cignoni, C. Rocchini, R. Scopigno*: Metro: Measuring error on simplified surfaces. Comput. Graph. Forum *17* (1998), 167–174.

[6] *P. Daniel, M. Medl'a, K. Mikula, M. Remešíková*: Reconstruction of surfaces from point clouds using a Lagrangian surface evolution model. Scale Space and Variational Methods in Computer Vision. Lecture Notes in Computer Science 9087. Springer, Cham, 2015, pp. 589–600.

[7] *S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, J. C. Hart*: Spectral surface quadrangulation. ACM Trans. Graph. *25* (2006), 1057–1066.

[8] *G. Dziuk*: Algorithm for evolutionary surfaces. Numer. Math. *58* (1991), 603–611.

[9] *G. Dziuk, C. M. Elliott*: Finite element methods for surface PDEs. Acta Numerica *22* (2013), 289–396.

[10] *C. M. Elliott, H. Fritz*: On approximations of the curve shortening flow and of the mean curvature flow based on the DeTurck trick. IMA J. Numer. Anal. *37* (2017), 543–603.

[11] *E. Faure, T. Savy, B. Rizzi, et al.*: A workflow to process 3D+time microscopy images of developing organisms and reconstruct their cell lineage. Nature Communications *7* (2016), Article ID 8674, 10 pages.

[12] *M. Fecko*: Differential Geometry and Lie Groups for Physicists. Cambridge University Press, Cambridge, 2006.

[13] *A. Gray*: Modern Differential Geometry of Curves and Surfaces with Mathematica. CRC Press, Boca Raton, 1998.

[14] *T. Y. Hou, I. Klapper, H. Si*: Removing the stiffness of curvature in computing 3-D filaments. J. Comput. Phys. *143* (1998), 628–664.

[15] *T. Y. Hou, J. S. Lowengrub, M. J. Shelley*: Removing the stiffness from interfacial flows with surface tension. J. Comput. Phys. *114* (1994), 312–338.

[16] *J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, H. Bao*: Spectral quadrangulation with orientation and alignment control. ACM Trans. Graph. *27* (2008), Article ID 147, 9 pages.

[17] *M. Huska, D. Lazzaro, S. Morigi*: Shape partitioning via L_p compressed modes. J. Math. Imaging Vis. *60* (2018), 1111–1131.

[18] *M. Kimura*: Numerical analysis for moving boundary problems using the boundary tracking method. Japan J. Ind. Appl. Math. *14* (1997), 373–398.

[19] *B. Kósa, J. Haličková-Brehovská, K. Mikula*: New efficient numerical method for 3D point cloud surface reconstruction by using level set methods. Proceedings of Equadiff 14. Slovak University of Technology, SPEKTRUM STU Publishing, Bratislava, 2017, pp. 387–396.

[20] *Y.-K. Lai, M. Jin, X. Xie, Y. He, J. Palacios, E. Zhang, S.-M. Hu, X. Gu*: Metric-driven RoSy field design and remeshing. IEEE Trans. Visualization Comput. Graph. *16* (2010), 95–108.

[21] *B. Lévy, Y. Liu*: L_p Centroidal Voronoi Tessellation and its applications. ACM Trans. Graph. *29* (2010), Article ID 119, 11 pages.

[22] *D. Liu, G. Xu, Q. Zhang*: A discrete scheme of Laplace-Beltrami operator and its convergence over quadrilateral meshes. Comput. Math. Appl. *55* (2008), 1081–1093.
[23] *M. Medl'a, K. Mikula*: Gaussian curvature based tangential redistribution of points on evolving surfaces. Proceedings of Equadiff 14. Slovak University of Technology, SPEKTRUM STU Publishing, Bratislava, 2017, pp. 255–264.
[24] *M. Meyer, M. Desbrun, P. Schröder, A. H. Barr*: Discrete differential-geometry operators for triangulated 2-manifolds. Visualization and Mathematics III. Springer, Berlin, 2003, pp. 35–57.
[25] *K. Mikula, M. Remešíková, P. Sarkoci, D. Ševčovič*: Manifold evolution with tangential redistribution of points. SIAM J. Sci. Comput. *36* (2014), A1384–A1414.
[26] *K. Mikula, D. Ševčovič*: Evolution of plane curves driven by a nonlinear function of curvature and anisotropy. SIAM J. Appl. Math. *61* (2001), 1473–1501.
[27] *K. Mikula, D. Ševčovič*: A direct method for solving an anisotropic mean curvature flow of plane curves with an external force. Math. Methods Appl. Sci. *27* (2004), 1545–1565.
[28] *S. Morigi*: Geometric surface evolution with tangential contribution. J. Comput. Appl. Math. *233* (2010), 1277–1287.
[29] *A. Myles, N. Pietroni, D. Kovacs, D. Zorin*: Feature-aligned T-meshes. ACM Trans. Graph. *29* (2010), Article ID 117, 11 pages.
[30] *T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, M. Wacker*: Compressed manifold modes for mesh processing. Comput. Graph. Forum *33* (2014), 35–44.
[31] *S. Osher, R. Fedkiw*: Level Set Methods and Dynamic Implicit Surfaces. Applied Mathematical Sciences 153. Springer, Berlin, 2003.
[32] *J. A. Sethian*: Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science. Cambridge Monographs on Applied and Computational Mathematics 3. Cambridge University Press, Cambridge, 1999.
[33] *D. Ševčovič, S. Yazaki*: Computational and qualitative aspects of motion of plane curves with a curvature adjusted tangential velocity. Math. Methods Appl. Sci. *35* (2012), 1784–1798.
[34] *G. L. G. Sleijpen, D. R. Fokkema*: BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum. ETNA, Electron. Trans. Numer. Anal. *1* (1993), 11–32.
[35] *M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, E. Puppo*: Practical quad mesh simplification. Comput. Graph. Forum *29* (2010), 407–418.
[36] *Y. Tong, P. Alliez, D. Cohen-Steiner, M. Desbrun*: Designing quadrangulations with discrete harmonic forms. Symposium on Geometry Processing, SGP '06. Eurographics Association, 2006, pp. 201–210.
[37] *F. Usai, M. Livesu, E. Puppo, M. Tarini, R. Scateni*: Extraction of the quad layout of a triangle mesh guided by its curve skeleton. ACM Trans. Graph. *35* (2015), Article ID 6, 13 pages.
[38] *J. Wenzel, M. Tarini, D. Panozzo, O. Sorkine-Hornung*: Instant field-aligned meshes. ACM Trans. Graph. *34* (2015), Article ID 189, 15 pages.
[39] *D.-M. Yan, B. Lévy, Y. Liu, F. Sun, W. Wang*: Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput. Graph. Forum *28* (2009), 1445–1454.
[40] *H.-K. Zhao*: A fast sweeping method for Eikonal equations. Math. Comput. *74* (2005), 603–627.
[41] *H.-K. Zhao, S. Osher, R. Fedkiw*: Fast surface reconstruction using the level set method. Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision. IEEE, Los Alamitos, 2001, pp. 194–201.

[42] *H.-K. Zhao, S. Osher, B. Merriman, M. Kang*: Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. Comput. Vis. Image Underst. *80* (2000), 295–314.

Authors' addresses: *Martin Húska*, Department of Mathematics, University of Bologna, Piazza di Porta San Donato, 5, 40126 Bologna, Italy, e-mail: `martin.huska@unibo.it`; *Matej Medl'a* (corresponding author), Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering STU in Bratislava, Radlinského 11, 810 05 Bratislava, Slovakia, e-mail: `medla@math.sk`; *Karol Mikula*, Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering STU in Bratislava, Radlinského 11, 810 05 Bratislava, Slovakia, e-mail: `mikula@math.sk`; *Serena Morigi*, Department of Mathematics, University of Bologna, Piazza di Porta San Donato, 5, 40126 Bologna, Italy, e-mail: `serena.morigi@unibo.it`.