Computers & Graphics 82 (2019) 152–162

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on SMI 2019

Generalized volumetric foliation from inverted viscous flow

David Cohen*, Mirela Ben-Chen

Technion - Israel Institute of Technology, Haifa, Israel

ARTICLE INFO

Article history:
Received 19 March 2019
Revised 20 May 2019
Accepted 21 May 2019
Available online 29 May 2019

Keywords:
Geometric flow
Foliations
Geometric processing
Volumetric mapping

ABSTRACT

We propose a controllable geometric flow that decomposes the interior volume of a triangular mesh into a collection of encapsulating layers, which we denote by a *generalized foliation*. For star-like genus zero surfaces we show that our formulation leads to a foliation of the volume with leaves that are closed genus zero surfaces, where the inner most leaves are spherical. Our method is based on the three-dimensional Hele-Shaw free-surface injection flow, which is applied to a conformally inverted domain. Every time iteration of the flow leads to a new free surface, which, after inversion, forms a foliation leaf of the input domain. Our approach is simple to implement, and versatile, as different foliations can be generated by modifying the injection point of the flow. We demonstrate the applicability of our method on a variety of shapes, including high-genus surfaces and collections of semantically similar shapes.

1. Introduction

In the last few decades, the field of geometric flows has seen great progress. As a consequence, these flows have emerged as an essential tool in diverse disciplines such as material science, geometry analysis, topology, quantum field theory and the solution of partial differential equations, among many others [1]. In this work, we propose a geometric flow for closed surfaces, which is based on the three-dimensional Hele-Shaw injection flow [2].

Our flow has three main properties: (1) If the surface remains a single component during the flow, then it experimentally converges to a *sphere*; (2) The free surfaces, at each time-step of the flow, form a collection of encapsulating layers, which yield a *generalized foliation*; (3) The flow is *controllable*, as the center of innermost spherical layer can be positioned at various points interior to the input surface. Hence, it is possible to influence the speed of the deformation of different parts of the surface, and the progress of the flow. In addition, our flow can handle *high-genus* surfaces, by allowing the surface topology to change during the flow, thus enabling the flow to progress beyond flow singularities.

Even though there exist other geometric flows with similar properties, such as: Ricci Flow [3], Mean Curvature Flow (MCF) [4], conformalized MCF [5], Yamabe Flow [6] and many others, to the best of our knowledge none of these possess all of the properties mentioned above. In other words, either these flows do not converge to a sphere, or they do not generate encapsulating layers, and most of them, if not all, are not controllable.

We show experimentally that our formulation generates a foliation of the input domain when the input surface is a star-like genus zero surface with respect to the injection point. Furthermore, we demonstrate our flow on a variety of surfaces, showing controllability, similar foliations for semantically similar shapes, and interesting generalized foliations of high-genus surfaces. These results can potentially be used for further geometric processing, such as computing volumetric maps, and transferring volumetric textures.

1.1. Related work

At the core of our method lies the three-dimensional Hele-Shaw flow with an injection singular point. The Computational Fluid Dynamics (CFD) literature is rich with analytical, numerical and experimental studies of the Hele-Shaw flow as well as other related flow types. A full review is beyond the scope of this work. Our literature review is thus limited to the most relevant results and methods related to our suggested method.

Geometric flows. Geometric flows are abundant in the mathematical literature, e.g. flows such as the Ricci flow [3], Willmore flow [7] and mean curvature flow [4], to mention just a few. In the geometry processing literature, corresponding *discrete* flows have been developed, where some examples include discrete Ricci flow [8], discrete Willmore flow [9], conformal Willmore flow [10] and conformalized mean curvature flow [5]. While all these flows converge to a sphere, they do not have the property that we require for generating a foliation, namely that the resulting surfaces are *encapsulating*. Furthermore, these flows are fully defined by the

* Corresponding author.
E-mail address: david.co@cs.technion.ac.il (D. Cohen).

https://doi.org/10.1016/j.cag.2019.05.015
0097-8493/

initial surface, leaving no room for controllability. Our flow, on the other hand, has both these properties.

Hele-Shaw flows. Our method is based on the three-dimensional model of the Hele-Shaw flow with an injection singular point. A mathematical treatment of *Hele-Shaw* flows from the point of view of geometric function theory and potential theory, including a complex analytic approach can be found in [11]. The model equations we use are based on Darcy's equations, which were derived from the Navier–Stokes equations via homogenization [12]. The same model, with suction instead of injection, is known to be unstable in some cases, yielding the *viscous fingering* phenomenon [13]. A two-dimensional computer graphics simulation of this model, both its stable and unstable versions and including two-phase flow and interactive control, was presented in [14]. Viscous fingering is closely related to other pattern formation phenomena such as bacterial growth and snowflake formation, among others, all of which are examples of *Laplacian growth*. A thorough investigation of this topic can be found in [15]. We focus on a specific setting of 3D Hele-Shaw flow, which leads to a geometric flow that has the properties that we require.

Foliations. The field of foliations has emerged as a distinct field with the publication by Ehresmann and Reeb [16]. Being a well established mathematical field, many introductory texts are available. A summarized introduction to the topic is given in [17], while a more recent and thorough introduction to the theory of foliations can be found in [18]. Another book presenting the basic concepts in the theory of foliations together with some more advanced topics such as aspects of the spectral theory for Riemannian foliations as well as applications of the heat equation method to Riemannian foliations appears in [19]. In the context of computer graphics, a recent work [20] has presented an algorithm for constructing foliations in a discrete setting which are then used for the bijective parametrization of two and three-dimensional objects, over canonical domains. In that work though, the authors handle foliations with one-dimensional leaves, i.e. any such foliation is a decomposition of a domain into disjoint curves. The volumetric bijective parameterization is then generated by the *transversal sections* of the the one-dimensional foliation. We, on the other hand, generate directly the *two-dimensional* leaves, of which the inner-most is a sphere. Our work can be considered complimentary as it provides an alternative solution using *two-dimensional* foliations which can spur further work on this topic.

1.2. Overview and motivation

The decomposition of a manifold into immersed sub-manifolds, namely *leaves*, is called a *foliation*. These *leaves* are of the same dimension, and "fit together nicely" (see [17,18] for a rigorous definition). We propose a method for computing the foliation of the interior of a triangular mesh $\mathcal{M}$ by leaves which are closed surfaces, using a *three dimensional Hele-Shaw flow* in a *conformally inverted domain*. Our algorithm is composed of the following steps (see Fig. 1):

1. Normalize the initial input mesh $\mathcal{M}$ such that it resides inside the unit sphere.
2. Conformally invert $\mathcal{M}$ using a Möbius inversion through the unit sphere to get $\tilde{\mathcal{M}}$.
3. Evolve $\tilde{\mathcal{M}}$ using a 3D Hele-Shaw flow until it converges to a sphere, leading to a series of surfaces $\tilde{\mathcal{M}}^n$.
4. Conformally invert $\tilde{\mathcal{M}}^n$ using a Möbius inversion through the unit sphere, to get the final foliation $\mathcal{M}^n$.

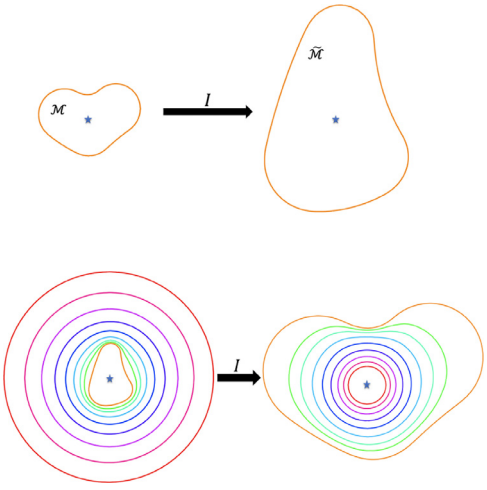

Fig. 1. Overview of our approach: (top) conformally invert the input mesh, (bottom, left part, zoomed out) then evolve the boundary using 3D Hele-Shaw flow, (bottom, right part) and finally invert back to get the foliation leaves. Every domain and its transformed domain have the same color. Origin is marked with $\star$. Injection point in the inverted domain, where the flow is executed, is at the origin.

The flow. The 3D Hele-Shaw flow that we use is a *normal flow*, where the normal velocity is determined by the gradient of a volumetric *harmonic function* which is 0 on the boundary of the domain, and negative in the interior. One important property of this flow, is that it is *positive by definition*, i.e. the inner product of the gradient of the harmonic function with the normal direction will be non-negative, hence, the domain that is occupied by the fluid at some instant *encapsulates* the domain occupied by the fluid in each of the previous times, leading to a foliation structure.

Further, our flow is derived from a physical model, where the existence of a solution globally in time was proven under the assumption that the initial domain is star-shaped with respect to the injection point [15, Theorem 4.5.2]. In addition, initial domains that are perturbations of balls converge to balls under this flow [21,22]. In our experiments, we demonstrate that even for initial domains that are not perturbations of balls, nor necessarily adhere to the star-likeness requirement, the resulting domain indeed converges to a ball.

The inversion map. The inversion map that we use maps the family of spheres centered at the origin to itself, so that a sphere is mapped to another sphere under this map, and the unit sphere remains unchanged. Furthermore, under this map, the interior of the unit sphere is mapped to be the exterior of the unit sphere, and vice-versa. Finally, this inversion map is conformal [23], hence the normal flow in the inverted domain is mapped to a normal flow in the original domain bounded by the initial input mesh. Since the inversion map is its own inverse, our method then finds the foliation leaves by applying the inversion map again on each of the resulting domains after each step of the viscous flow evolution.

2. Method

2.1. Notations

I	inversion map, a Möbius transformation.
$\mathcal{M}(\mathcal{V}_\mathcal{M}, \mathcal{F}_\mathcal{M})$	triangular surface mesh, faces are triangles.
$\mathcal{T}(\mathcal{V}_\mathcal{T}, \mathcal{F}_\mathcal{T})$	tetrahedral volumetric mesh, elements are tetrahedrons.
$\tilde{\mathcal{M}}$	triangular surface mesh after applying inversion I.

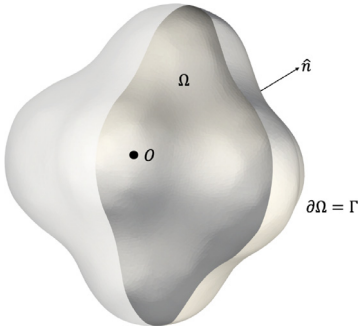

Fig. 2. The model geometry. The domain Ω with its boundary $\partial\Omega = \Gamma$, O is the singularity, placed at the origin. Half of the interior of the model is transparent for visualization.

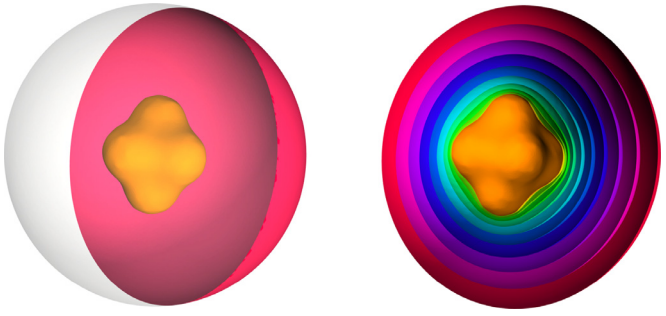

Fig. 3. The evolution of a star-like domain. (left) The input surface (orange) together with the final surface (red), which is a sphere. (right) The full injection flow, where every layer appears in a different color. We show the cross-section of 10 sampled layers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$\tilde{\mathcal{T}}$	the mesh generated after tetrahedralizing the volume whose boundary surface is $\tilde{\mathcal{M}}$.
$\partial\tilde{\mathcal{T}}$	the triangular mesh which is the boundary of the tetrahedral mesh $\tilde{\mathcal{T}}$.
$L_{\tilde{\mathcal{T}}} \in \mathbb{R}^{\lvert\mathcal{V}_{\tilde{\mathcal{T}}}\rvert \times \lvert\mathcal{V}_{\tilde{\mathcal{T}}}\rvert}$	the Laplacian operator matrix of the tetrahedral mesh $\tilde{\mathcal{T}}$.
$L_{\tilde{\mathcal{M}}} \in \mathbb{R}^{\lvert\mathcal{V}_{\tilde{\mathcal{M}}}\rvert \times \lvert\mathcal{V}_{\tilde{\mathcal{M}}}\rvert}$	the Laplacian operator matrix of the triangular mesh $\tilde{\mathcal{M}}$.
$G_{\tilde{\mathcal{T}}} \in \mathbb{R}^{3\lvert\mathcal{F}_{\tilde{\mathcal{T}}}\rvert \times \lvert\mathcal{V}_{\tilde{\mathcal{T}}}\rvert}$	the gradient operator matrix of the tetrahedral mesh $\tilde{\mathcal{T}}$.
$G_{\tilde{\mathcal{M}}} \in \mathbb{R}^{3\lvert\mathcal{F}_{\tilde{\mathcal{M}}}\rvert \times \lvert\mathcal{V}_{\tilde{\mathcal{M}}}\rvert}$	the gradient operator matrix of the triangular mesh $\tilde{\mathcal{M}}$.

Our goal is to compute a series of encapsulating surfaces in the interior of our input domain $\mathcal{M}$. Since injection flow is stable and converges to a ball, we first *invert* the input mesh through the unit sphere, then run the flow to get the layers, and then invert back.

2.2. The flow

We assume the model of the Hele-Shaw problem in $\mathbb{R}^3$. We consider the evolution of an incompressible three-dimensional viscous fluid, under the influence of injection through a singular point x_0 internal to the fluid domain $\tilde{\Omega}$. The velocity is divergence free everywhere except at the singular point x_0. See Fig. 2 for an illustration of the model geometry. We follow the model Eqs. (1.1)–(1.3) from [2], with two minor modifications. First, we define the injection rate with an opposite sign, and second, we denote the pressure as the velocity potential $\Phi : \Omega \to \mathbb{R}$, so that the velocity u is defined by,

$$u = -\nabla\Phi. \tag{1}$$

We get:

$$\Delta\Phi = Q\delta_{x_0}(x), \qquad x, x_0 \in \tilde{\Omega}(t) \tag{2}$$

$$\Phi = \sigma H, \qquad on\ \tilde{\Gamma}(t), \tag{3}$$

where Δ is the Laplacian, Q a constant indicating a rate of injection ($Q<0$ for a source) or suction ($Q>0$ for a sink), $\delta_{x_0}(x)$ is the three-dimensional Dirac distribution centered at x_0, $\tilde{\Omega} \subset \mathbb{R}^3$ is the fluid domain, $\partial\tilde{\Omega} = \tilde{\Gamma}$ is the domain boundary, σ is the surface tension and H is the mean curvature. The solution for Φ determines the velocity of the boundary:

$$u_n = \langle -\nabla\Phi(x), \hat{n}(x)\rangle \qquad x \in \tilde{\Gamma}(t) \tag{4}$$

where $\hat{n}(x)$ is the outward unit normal direction to the boundary $\tilde{\Gamma}$ and u_n is the normal component of the velocity at the boundary $\tilde{\Gamma}$. A solution to the partial differential equation in (2) is of the form:

$$\Phi = QG(x, x_0) + g(x) \qquad x, x_0 \in \tilde{\Omega} \tag{5}$$

where $G(x, x_0)$ is the Green's function for the Laplacian in $\mathbb{R}^3$, given by $G(x, x_0) = -\frac{1}{4\pi|x-x_0|}$, and $g(x)$ is a harmonic function necessary for fulfilling the boundary conditions. Also, for the purpose of our method we assume a flow with a constant rate of injection $Q<0$ and without surface tension. Hence, the following are our model equations,

$$\Phi(x) = QG(x, x_0) + g(x) \qquad x \in \tilde{\Omega} \tag{6}$$

$$\Phi(x) = 0 \qquad x \in \tilde{\Gamma} \tag{7}$$

$$u_n = \langle -\nabla\Phi(x), \hat{n}(x)\rangle \qquad x \in \tilde{\Gamma} \tag{8}$$

This derivation is similar to the derivation presented in [11] which was also used in [14] to model the one-phase two-dimensional *Hele-Shaw* flow. We refer the reader to [2,11] as well as [15] for further details.

Since we consider a viscous flow with injection (i.e. $Q<0$), the potential Φ in all the domain $\tilde{\Omega}$ is *positive* and the velocity at the boundary $\tilde{\Gamma}$ points *outwards* such that the boundary expands. Fig. 3 shows an example of the evolution of a star-like domain which shows the expanding boundary.

2.3. The inversion map

Given an input initial triangular mesh $\mathcal{M}$, which we consider as the boundary of an initial domain $\partial\Omega(0) = \Gamma(0)$, our goal is to efficiently find a family of domains $\tilde{\Omega}(t)$ which fulfill the model Eqs. (6)–(8). $\tilde{\Omega}(0)$ is the domain bounded by the inverted input mesh $\tilde{\mathcal{M}}$, i.e. the domain obtained after applying a Möbius transform $I : x \mapsto \tilde{x}, \quad x, \tilde{x} \in \mathbb{R}^3$ to the given input mesh $\mathcal{M}$ as follows,

$$\tilde{x} = \begin{cases} x/||x||^2 & \text{if } x \neq 0, \infty \\ 0 & \text{if } x = \infty \\ \infty & \text{if } x = 0 \end{cases} \tag{9}$$

The map I is the inversion of $\mathbb{R}^3 \cup \{\infty\}$ with respect to the unit sphere, such that for $x \notin \{0, \infty\}$, $\tilde{x}$ is on the ray with an endpoint at the origin which passes through x, with $|\tilde{x}| = 1/|x|$. It is easy to see that the map is continuous and is its own inverse. Once the family of domains $\tilde{\Omega}(t)$ is obtained, the volumetric foliation of the volume bounded by the initial input mesh $\mathcal{M}$ is computed by applying the map I to each domain and taking its boundary, i.e. the foliation leaves are $\partial\Omega(t)$. Fig. 4 shows an example of the application of the inversion map.

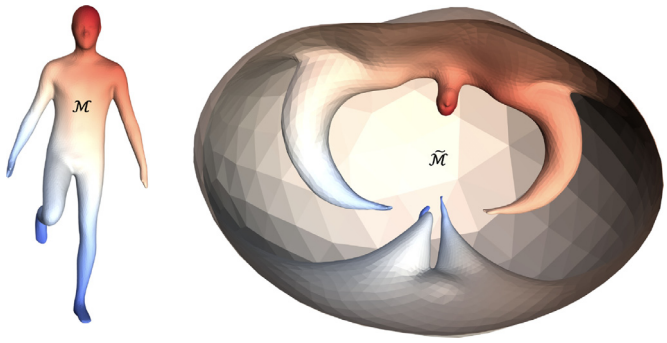

Fig. 4. Example of the inversion map, the meshes are color coded such that corresponding points have the same color. (left) The original mesh. (right) An inversion of the standing man mesh.

2.4. Discretization

For solving model Eqs. (6)–(8), we take the inverted initial input mesh $\tilde{\mathcal{M}}(\mathcal{V}_{\tilde{\mathcal{M}}}, \mathcal{F}_{\tilde{\mathcal{M}}})$ which we refer to as $\partial\tilde{\Omega}(0)$, and discretize the domain it bounds, i.e. $\tilde{\Omega}(0)$, using a tetrahedral mesh $\tilde{\mathcal{T}}(\mathcal{V}_{\tilde{\mathcal{T}}}, \mathcal{F}_{\tilde{\mathcal{T}}})$. We denote by $\partial\tilde{\mathcal{T}}(\mathcal{V}_{\partial\tilde{\mathcal{T}}}, \mathcal{F}_{\partial\tilde{\mathcal{T}}})$ the triangular mesh which is the boundary of $\tilde{\mathcal{T}}$, which should be exactly $\tilde{\mathcal{M}}$, but as tetrahedralization procedures tend to generate meshes with boundaries that do not perfectly fit the mesh that was used to define their boundary, we will need to refer to this surface as well. Solving the model means finding the harmonic $g(x)$, as implied from Eq. (2). We chose to work with FEM as our discretization, hence we consider a scalar function as a piecewise linear function defined over the vertices. Therefore, we seek a discretized harmonic function g which satisfies

$$L_{\tilde{\mathcal{T}}} g = 0, \tag{10}$$

where $L_{\tilde{\mathcal{T}}}$ is the Laplacian operator matrix defined for a tetrahedral mesh $\tilde{\mathcal{T}}$. Writing our discretized model (10), including the boundary conditions, in matrix form we get

$$\left[\begin{array}{c|c} L_{\tilde{\mathcal{T}}_{\mathcal{I}\times\mathcal{I}}} & L_{\tilde{\mathcal{T}}_{\mathcal{I}\times\mathcal{B}}} \\ \hline 0 & I_{\mathcal{B}} \end{array}\right]\left[\begin{array}{c} X_{\mathcal{I}} \\ \hline X_{\mathcal{B}} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline g_{\mathcal{B}} \end{array}\right], \tag{11}$$

where $\mathcal{I}$ and $\mathcal{B}$ are the sets of indices of the interior and boundary vertices, respectively, and $I_{\mathcal{B}}$ is the identity matrix of the appropriate size. We evaluate the boundary values $g_{\mathcal{B}} = g\big|_{\partial\tilde{\mathcal{T}}}$ using Eqs. (6) and (7),

$$g_{\mathcal{B}} = -Q \cdot \mathcal{G}, \tag{12}$$

where $\mathcal{G}$ is the vector of Green's function $G(v, v_0)$ values evaluated at every vertex $v \in \partial\tilde{\mathcal{T}}$. Since $g_{\mathcal{I}}$ is what we seek, the system is simplified to be,

$$[L_{\tilde{\mathcal{T}}_{\mathcal{I}\times\mathcal{I}}}][X_{\mathcal{I}}] = [-L_{\tilde{\mathcal{T}}_{\mathcal{I}\times\mathcal{B}}} \cdot g_{\mathcal{B}}] \tag{13}$$

which gives the values of $g_{\mathcal{I}}$, and together with the values $g_{\mathcal{B}}$ obtained in (12) we have the values of the discretized harmonic function g for all $v \in \mathcal{V}_{\tilde{\mathcal{T}}}$. Using the discrete gradient operator matrix defined for a tetrahedron-mesh, we can calculate $\mathbf{G}^g_{\mathcal{F}_{\tilde{\mathcal{T}}}} = G_{\tilde{\mathcal{T}}} g$ defined over the tetrahedrons $t \in \mathcal{F}_{\tilde{\mathcal{T}}}$.

2.5. Interpolating ∇g to Vertices of $\partial\tilde{\mathcal{T}}$

We use the *barycentric volume* (adapted from *barycentric area* defined in [24]) in order to interpolate $\mathbf{G}^g_{\mathcal{F}_{\tilde{\mathcal{T}}}}$ values from the tetrahedrons to the mesh vertices. Denoting $Vol_{\mathcal{F}}(t_j), t_j \in \mathcal{F}_{\tilde{\mathcal{T}}}$ as the tetrahedral volume of t_j, then $Vol_{\mathcal{V}}(v_i) = \sum_{t_j \in N_{1_i}} Vol_F(t_j)/4$ is the vertex volume of v_i, where N_{1_i} is vertex v_i 1−ring neighborhood of tetrahedrons. We define an interpolation operator $I^{\mathcal{F}}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}_{\tilde{\mathcal{T}}}|\times|\mathcal{F}_{\tilde{\mathcal{T}}}|}$ to interpolate values from the tetrahedrons to the vertices, in order to get the values $\mathbf{G}^g_{\mathcal{V}_{\tilde{\mathcal{T}}}}$ defined over the vertices $v \in \mathcal{V}_{\tilde{\mathcal{T}}}$. Practically, $I^{\mathcal{F}}_{\mathcal{V}}(i,j) = \frac{Vol_{\mathcal{F}}(t_j)}{4Vol_{\mathcal{V}}(v_i)}$ iff face j belongs to the 1−ring neighborhood of vertex i and 0 otherwise, so that,

$$\mathbf{G}^g_{\mathcal{V}_{\tilde{\mathcal{T}}}} = I^{\mathcal{F}}_{\mathcal{V}} \cdot \mathbf{G}^g_{\mathcal{F}_{\tilde{\mathcal{T}}}} \tag{14}$$

2.6. Interpolating velocity to vertices of $\tilde{\mathcal{M}}$

Having $\mathbf{G}^g_{\mathcal{V}_{\tilde{\mathcal{T}}}}$, we can easily calculate the gradient of the discretized potential defined over the vertices $v \in \mathcal{V}_{\tilde{\mathcal{T}}}$, denoted $\mathbf{G}^{\phi}_{\mathcal{V}_{\tilde{\mathcal{T}}}}$. Though, what we would actually like to find are the values $\mathbf{G}^{\phi}_{\mathcal{V}_{\tilde{\mathcal{M}}}}$ - the gradient values of the discretized potential defined over the vertices $v \in \mathcal{V}_{\tilde{\mathcal{M}}}$, since this is our surface mesh of interest. Finding the velocity of those vertices is then merely $u = -\mathbf{G}^{\phi}_{\mathcal{V}_{\tilde{\mathcal{M}}}}$, as Eq. (1) suggests. Though, for finding $\mathbf{G}^{\phi}_{\mathcal{V}_{\tilde{\mathcal{M}}}}$ we project the vertices that belong to the boundary surface of $\tilde{\mathcal{T}}$, i.e. $v \in \partial\tilde{\mathcal{T}}$, on the triangular surface $\tilde{\mathcal{M}}$. We denote those projected vertices positions by P. We have $\forall p \in P, \exists f \in \mathcal{F}_{\tilde{\mathcal{M}}}$ s.t. p is on the plane defined by f (the positions p are somewhere on the faces of $\tilde{\mathcal{M}}$). As a side note we will mention that a tetrahedral mesh can be generated in such a way that it can be guaranteed that the vertices that belong to its boundary surface are on the faces of the mesh used as an input for the generation process, meaning the projection step described above is not needed. Nevertheless, as we do not generate a tetrahedral mesh every step for a better performance, but rather evolve it as well for a few iterations, therefore this projection step can not be discarded. Refer to Section 2.8 for further details. We assume the gradient values of the discretized potential evaluated at the vertices $v \in \partial\tilde{\mathcal{T}}$ are the same as the values evaluated at the positions P after projection, denoted $\mathbf{G}^{\phi}_{P}$. Defining $B \in \mathbb{R}^{|P|\times|\mathcal{V}_{\tilde{\mathcal{M}}}|}$ as the matrix of barycentric coordinates of the projected positions p relative to the faces $\mathcal{F}_{\tilde{\mathcal{M}}}$, and solving, in the Least Squares (LS) sense, the regularized system,

$$\left[\begin{array}{c} B \\ \hline \epsilon_1 G_{\tilde{\mathcal{M}}} \end{array}\right][X]_{|\mathcal{V}_{\tilde{\mathcal{M}}}|\times 1} = \left[\begin{array}{c} \mathbf{G}^{\phi}_{P} \\ \hline 0 \end{array}\right] \tag{15}$$

gives the desired values of $\mathbf{G}^{\phi}_{\mathcal{V}_{\tilde{\mathcal{M}}}}$, where $G_{\tilde{\mathcal{M}}}$ is the gradient operator matrix of the surface mesh $\tilde{\mathcal{M}}$ and ϵ_1 is a factor determining how effective the regularization will be.

2.7. Evolving the surface boundary

As noted earlier, applying Eq. (1) we get the velocities $u = -\mathbf{G}^{\phi}_{\mathcal{V}_{\tilde{\mathcal{M}}}}$ of the vertices $v \in \mathcal{V}_{\tilde{\mathcal{M}}}$, and we can evolve the surface boundary mesh $\tilde{\mathcal{M}}$. We project the velocity u of each vertex $v \in \mathcal{V}_{\tilde{\mathcal{M}}}$ on the appropriate surface normal at each vertex to get the normal velocities u_n. Finding the new locations of the vertices $v \in \mathcal{V}_{\tilde{\mathcal{M}}}$ is then being done by solving the system, in the LS sense,

$$\left[\begin{array}{c} I_{n_{\tilde{\mathcal{M}}}} \\ \hline \epsilon_2 \mu L_{\tilde{\mathcal{M}}} \end{array}\right][X] = \left[\begin{array}{c} \bar{v} + \partial\tau \cdot u_n \\ \hline 0 \end{array}\right]. \tag{16}$$

Here $I_{n_{\tilde{\mathcal{M}}}}$ is the identity matrix of the appropriate size, $L_{\tilde{\mathcal{M}}}$ is the Laplacian operator matrix of $\tilde{\mathcal{M}}$, $\partial\tau$ is the time interval length, $\bar{v}$ being the vector of the vertices $v \in \mathcal{V}_{\tilde{\mathcal{M}}}$, μ being the smallest eigenvalue (which is not 0) of $L_{\tilde{\mathcal{M}}}$ that acts as a regularizer, and ϵ_2 is a factor determining how effective the regularization will be. Denoting the resulting set of vertices as $\mathcal{V}^{\tau+\partial\tau}_{\tilde{\mathcal{M}}}$, the mesh defined by $(\mathcal{V}^{\tau+\partial\tau}_{\tilde{\mathcal{M}}}, \mathcal{F}_{\tilde{\mathcal{M}}})$ is set to be the surface mesh $\tilde{\mathcal{M}}^n(\mathcal{V}^n_{\tilde{\mathcal{M}}}, \mathcal{F}^n_{\tilde{\mathcal{M}}})$ which is the boundary of the domain for the next iteration. Once in a few iterations the mesh defined by $(\mathcal{V}^{\tau+\partial\tau}_{\tilde{\mathcal{M}}}, \mathcal{F}_{\tilde{\mathcal{M}}})$ is first being remeshed

and only then set to be $\tilde{\mathcal{M}}^n(\mathcal{V}^n_{\tilde{\mathcal{M}}}, \mathcal{F}^n_{\tilde{\mathcal{M}}})$. The current *foliation leaf* is the surface mesh defined by the faces $\mathcal{F}^n_{\tilde{\mathcal{M}}}$ and by the vertices obtained by applying the inversion map (9) to the vertices $\mathcal{V}^n_{\tilde{\mathcal{M}}}$. In cases where we want to have all the foliation leaves with the same triangulation as the original manifold, then a post process operation of iteratively projecting on each leaf, from outermost to innermost, the set of vertices projected on the previous leaf, where initially this set is $\mathcal{V}_{\tilde{\mathcal{M}}}$. This processing can also be done in parallel to the evolution, at the end of each step, and not necessarily as a post process step.

2.8. Reusing tetrahedral mesh $\tilde{\mathcal{T}}$

In order to speed up execution time, we reduce the number of tetrahedral mesh generations and reuse results from previous iterations. Since we know $\mathbf{G}^g_{\mathcal{V}_{\partial\tilde{\mathcal{T}}}}$ for the vertices on the boundary surface of the mesh $\partial\tilde{\mathcal{T}}$, we therefore can calculate the velocity of those vertices, denoted by $w_{\mathcal{B}}$, and we can evolve the boundary of the tetrahedral mesh. As for updating the locations of the interior vertices of $\tilde{\mathcal{T}}$, since we do not need those vertices to move according to the model Eqs. (6)–(8) we rather use the velocity of the boundary vertices and find a smooth solution for the interior vertices, i.e. solve in the LS sense the following system,

$$\left[\begin{array}{c|c} L_{\mathcal{I}\times\mathcal{I}} & L_{\mathcal{I}\times\mathcal{B}} \\ \hline 0 & I_{\mathcal{B}\times\mathcal{B}} \end{array}\right]\left[\begin{array}{c} X_{\mathcal{I}} \\ \hline X_{\mathcal{B}} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline w_{\mathcal{B}} \end{array}\right] \tag{17}$$

Similar to before, simplifying of the system yields,

$$[L_{\tilde{\mathcal{T}}_{\mathcal{I}\times\mathcal{I}}}][X_{\mathcal{I}}] = [-L_{\tilde{\mathcal{T}}_{\mathcal{I}\times\mathcal{B}}} \cdot w_{\mathcal{B}}] \tag{18}$$

and we get $w_{\mathcal{I}}$. With the complete velocity vector w for the vertices of the tetrahedral mesh we then advance the vertices $\mathcal{V}_{\tilde{\mathcal{T}}}$ locations by $\partial\tau \cdot w$ to get their new locations. We still regenerate a new tetrahedral mesh but only once in a few iterations.

2.9. High genus

Our method is capable of handling high genus meshes without boundaries, if we allow the flow to change the topology of the evolved surface by continuing beyond a flow singularity, when such occurs. Our method handles flow singularities by using the following rather simple heuristic:

- Find the number of *degenerate* triangular faces (see below for further details), let's call this number d.
- If $d > t_1$, where t_1 is the minimal number of degenerate triangles that we would like to remove at once, then:
  - If d has increased from the previous evolution iteration, and $d < t_2$ then continue, where t_2 is the maximal number of degenerate triangles that allowed to exist unhandled.
  - If $d > t_2$, or d has remained the same as in the previous evolution iteration (but $d > t_1$), then all the degenerate triangles are being marked to be removed from the surface.
- Apply removal of faces marked to be removed, only if the genus after the removal will decrease.

Whenever such faces are removed, our method executes a holes filling procedure to fill up the holes that were created due the faces removal, so that the resulted mesh is again without boundaries. The evolution continues with this new surface of lower order genus, until a convergence to a sphere is reached.

We tried two methods to determine whether a triangular face is considered *degenerate* for the heuristic above:

- A face whose area is less than a threshold θ, where $\theta \ll 1$.
- A face whose Shape Diameter Function (SDF) value is less than a threshold μ, where $\mu < 1$.

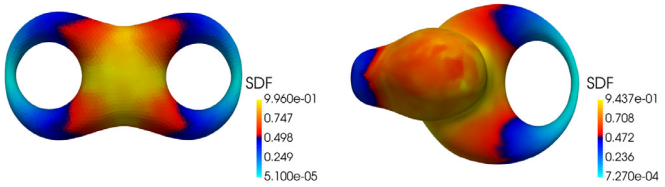

Fig. 5. SDF values for different meshes. (left) Eight. (right) Bob the duck.

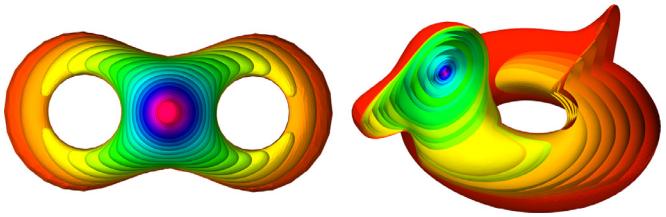

Fig. 6. High genus. Result obtained after allowing the flow to progress beyond the flow singularities, with the heuristic described at Section 2.9.

The SDF, as appeared in [25], is a scalar function defined on the mesh surface. It expresses an estimate of the diameter of the object's volume in the neighborhood of each point on the surface. The SDF values remain almost the same for different poses of the same object. Fig. 5 shows examples for SDF values of two meshes, after a few evaluations of our flow, and before reaching a flow singularity. It is noticeable in both of the examples that the areas where flow singularity is about to occur are areas of low SDF values. In our tests we used values of $\theta \in (1e{-}7, 1e{-}5)$ and $\mu \in (5e{-}3, 2e{-}2)$. Although the method that designates *degenerate* faces according to their area is simpler to implement, the method that uses the SDF values seems to perform better, and operates as expected on more high-genus objects. We used the implementation of the SDF values calculation provided as part of CGAL [26]. Figs. 6 and 19 show results of our method when running on high-genus meshes.

3. Implementation details

In our experiments we executed our method on surfaces that are bounded by the unit sphere. That way, after applying the inversion, the inverted surface is then completely outside the unit sphere. Working in that manner is not obligatory, but rather it helped in keeping each parameter used in our method of the same order of magnitude when running on different surfaces. In the following subsections we will describe the parameters we used when executing our method and how we set their values.

3.1. Time step interval $\partial\tau$ and injection rate Q

In our experiment we kept the time interval constant and used the value of $\partial\tau = 0.1$. Since the evolution happens in the inverted domain, and is executed as a flow of injection from a singular point, therefore as the boundary expands at every iteration of the simulation, the injection rate affects less and less, since the distance from the singularity to the boundary increases. In other words, the relative progress of the simulation is reduced along the execution. This behavior can be demonstrated when considering the evolution of a sphere according to our model. It can easily be shown that when the initial domain is a sphere, then when applying our model Eqs. (6)–(8) in that domain gives rise to the following equation for the radius of the sphere,

$$R(t) = \sqrt[3]{R_0^3 - \frac{3Qt}{4\pi}} \tag{19}$$

which supports our claim on the slowing in the simulation progress. Reminder: we handle a flow of injection, i.e. $Q < 0$. See Appendix A for a derivation of Eq. (19). In order to tackle this problem, we use a heuristic for adaptively changing the value of the injection rate Q. The main idea is that we have a predefined parameter α determining the requested movement of the closest point to the singularity in percentage from its distance to the singularity. Notice that the closest point to the singularity, denoted by ρ, can change between iterations. In other words, we would like the movement of ρ to be of size $\alpha \cdot \rho$. At every iteration we check the actual movement that we achieved, denoted $\partial\rho$. According to the difference $(\alpha \cdot \rho - \partial\rho)$, we increase or decrease (by a constant factor) yet another parameter γ, where if the current time is τ then γ should take the value as in the following relation $(\gamma\rho(\tau))^3 = \rho(\tau - \partial\tau)^3 - \rho(\tau)^3$. The new value of Q is then (in the spirit of Eq. (19), as if the domain is spherical),

$$Q = \frac{\gamma^3 \cdot 4\pi}{3 \cdot \partial\tau} \tag{20}$$

3.2. Triangular and tetrahedral meshes handling

In our implementation we chose to use the Finite Element Method (FEM). Alternative approaches can be taken for implementing the method described in Section 2. For instance, in [27] the authors used the Boundary Element Method (BEM) for addressing a model closely related to the model we use for simulating the 3D Hele-Shaw flow. In our experiments we also tried to utilize a method suggested in [28] which is another BEM alternative for solving our model. However, since we did not experience any performance benefit, we decided to choose FEM. For discretizing the volumetric domain, we tried using two alternatives, CGAL [26] and TetWild [29]. We used TetWild while it was still in development, and though it gave results that seem to be better for our needs (for example, controlling the deviation of the generated tetrahedral mesh boundary surface from the triangular surface that was given as an input is much easier than CGAL), we eventually chose to use CGAL for tetrahedral mesh generation due its better performance, and since it is easier to work with it as it has a C/C++ API, while TetWild is black box utility. We also used CGAL capabilities for triangular mesh remeshing and hole filling. In our implementation, after evolving the boundary and applying the inversion map I, we remesh the resulting surface to get the current foliation leaf. As for the tetrahedral mesh generation, we execute it once in every 20 iterations, while in the other iterations we update the tetrahedral mesh vertices as described in Section 2.8.

3.3. Regularization factors

3.3.1. ϵ_1

The purpose of ϵ_1 is to prevent the system in Eq. (15) from being rank deficient. As we would like to interpolate values from the projected positions P which are on the faces $\mathcal{F}_{\tilde{\mathcal{M}}}$, to the vertices $\mathcal{V}_{\tilde{\mathcal{M}}}$, it is not guaranteed that for every face $f \in \mathcal{F}_{\tilde{\mathcal{M}}}$ there exists a point $p \in P$ which is on the face f. Therefore, there might be vertices which all the faces they belong to have no points in P placed on them. Meaning, no interpolation of the velocity can be done for those vertices when solving Eq. (15). That is where the regularization factor ϵ_1 comes in, as it causes to the velocity to be smooth. Fig. 7 shows how ϵ_1 affects the flow. The value of ϵ_1 needs not to be too big as it causes aggressive smoothing to the velocity of the boundary mesh vertices, which slows down the evolution. Furthermore, it can be seen that for larger values of ϵ_1 features (such as the hands, legs, ears, etc.) disappear at a later stage in the flow. We should note that the result on the right in Fig. 7, that shows an evolution with larger ϵ_1 value, does not converge to a sphere only because the simulation was halted, due to very long running time.

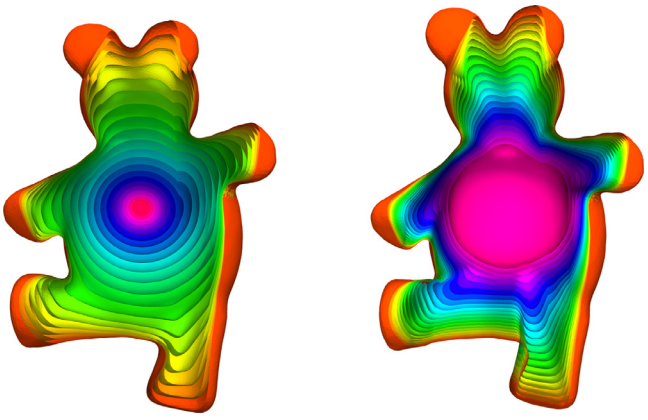

Fig. 7. ϵ_1 effectiveness demonstrated on a teddy mesh foliation result. Meshes are clipped using 3 planes. (left) $\epsilon_1 = 0.09$. (right) $\epsilon_1 = 0.9$.

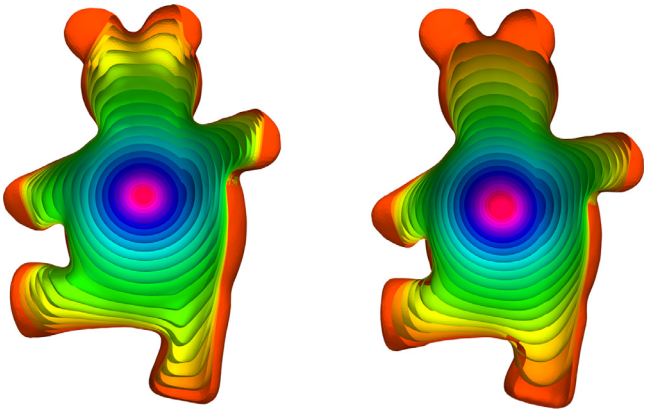

Fig. 8. ϵ_2 effectiveness demonstrated on a teddy mesh foliation result. (left) $\epsilon_2 = 6e{-}7$. (right) $\epsilon_2 = 6e{-}5$.

We found that the value of $\epsilon = 0.09$ works fine for the desired results, and used it in all of our experiments.

3.3.2. ϵ_2

The purpose of ϵ_2 is to smooth the new positions of the vertices $\mathcal{V}_{\tilde{\mathcal{M}}}$. Having this factor helps to obtain better evolution results. A value too big of ϵ_2 is problematic though. In Fig. 8, the result on the right, which matches an evolution with a rather large value of ϵ_2, demonstrates the problems with a smoothing of the new positions which is too aggressive. Besides having the features (such as the hands, legs, ears, etc.) taking spherical shapes, the bigger problem is that it causes intersections between the layers, as can be clearly seen happens in the right leg. We used the value of $\epsilon_2 = 6e{-}7$ in all of our experiments.

3.4. Limitations

3.4.1. Flow singularities

As indicated already in Section 1.2, the Hele-Shaw flow with injection is in principle well posed. Though, when requiring a solution that is not allowed to change topology, a flow singularity will be encountered for an initial geometry sufficiently entangled (see [15], Chapter 4 Section 4.2 and Theorem 4.5.2). High-genus meshes are a typical example for such a geometry where the flow will reach a singularity. Such singularities, however, can also arise in from genus zero meshes as well. See Fig. 9 for such an example.

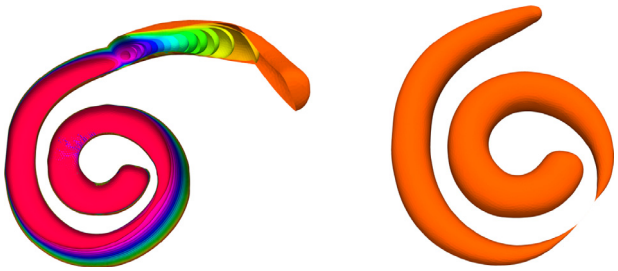

Fig. 9. Spring mesh, genus-zero failure case. (left) Layers obtained until flow singularity reached. (right) The mesh in time of flow singularity. The mesh is a one component mesh with degenerate triangles.

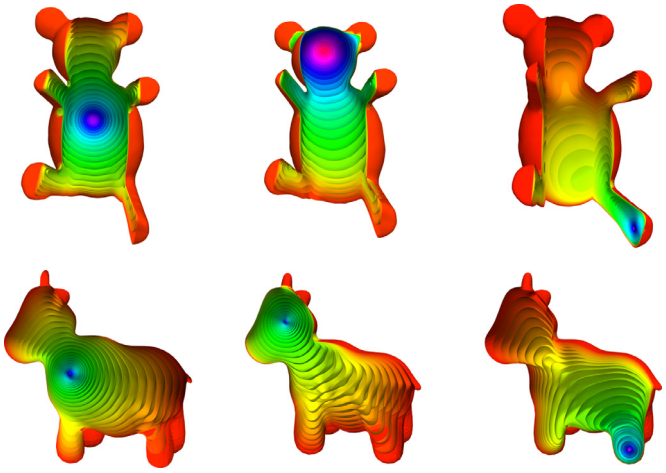

Fig. 10. Controlling the flow. Showing the same mesh with different singular points.

3.4.2. Layer intersections

Results obtained using our implementation might encounter intersections of the layers, that should be non-intersecting in general. Two typical reasons for this to happen:

- The parameter ϵ_2, described in Section 3.3.2 in detail, might cause such layer intersections if not tuned properly, as demonstrated in Fig. 8.
- An inherent flaw in our implementation due to relying on remeshes. In general, our method remeshes the currently evolved mesh after the new positions for the vertices were calculated.
  - In areas of low to no velocity such a remesh means essentially the effective velocity would differ from the original one not only in magnitude, but even in its direction. Therefore, areas that are almost static in a rather long period of the evolution might suffer intersections of their appropriate layers.
  - In a similar manner, areas of high curvature are more likely to suffer intersections of their appropriate layers, as easily can be seen in our results for high-genus meshes, in areas forming the holes.

4. Experimental results

We demonstrate our method on various genus-zero meshes and on higher-genus meshes as well. In Figs. 15–17 we show the results of our method applied on inputs of closely related meshes, with the singular point being located in similar positions along the meshes. In Figs. 20–22 we show results after applying our method on inputs of the same mesh at different poses. In both of these sets of results it is noticeable that similar meshes produce similar layers, or phrasing it differently, the convergence to a sphere looks similar for similar meshes. In Fig. 19 we show some more results of our method when applied on high-genus meshes. In Fig. 14 we demonstrate the use of the texture of the object. We show results for 3 meshes and present only the initial mesh given as an input and the final mesh obtained at the end of the evolution, though all the intermediate surface meshes from all the evolution steps have a texture. We use the same texture coordinates of the initial mesh given as an input for all the layers obtained during the evolution, meaning we need the same triangulation for all the layers, which is accomplished by the method described in Section 2.7.

4.1. Controlling the flow

One of the novelties of our method is that it introduces a flow that is controllable, and this control is achieved by choosing the location of the injection point, when simulating the 3D Hele-Shaw flow in the inverted domain. Choosing a location that the domain is starshaped with respect to it, results in a flow that is guaranteed to converge, but as our results show, convergence can be achieved even for non starshaped configurations. In our implementation, we chose to always use the origin as the injection point, and translated the domain according to the desired location of the

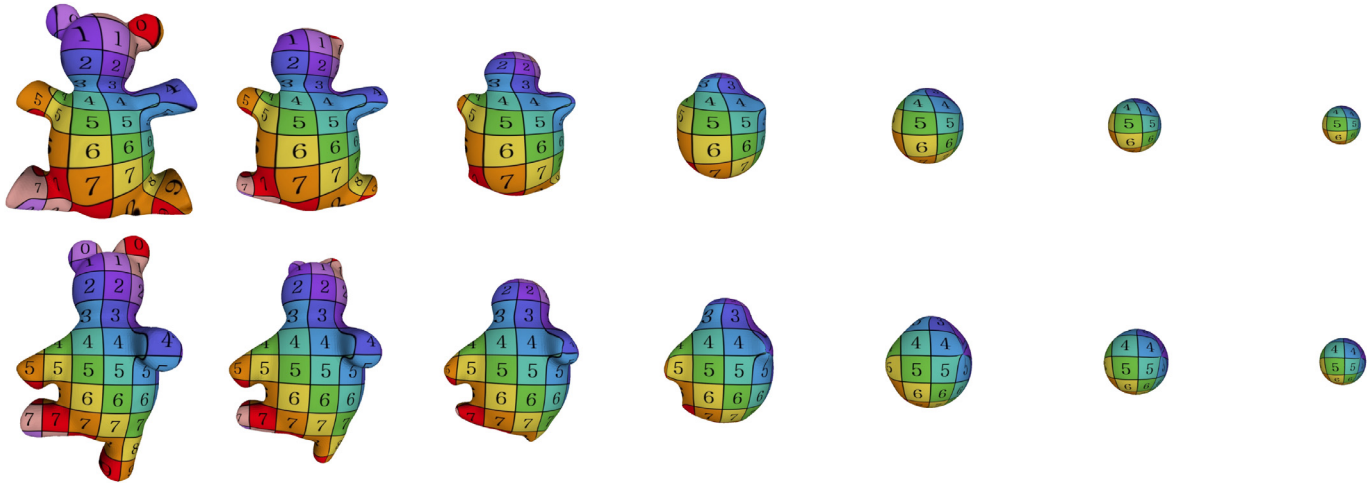

Fig. 11. Teddy meshes volume mapping.

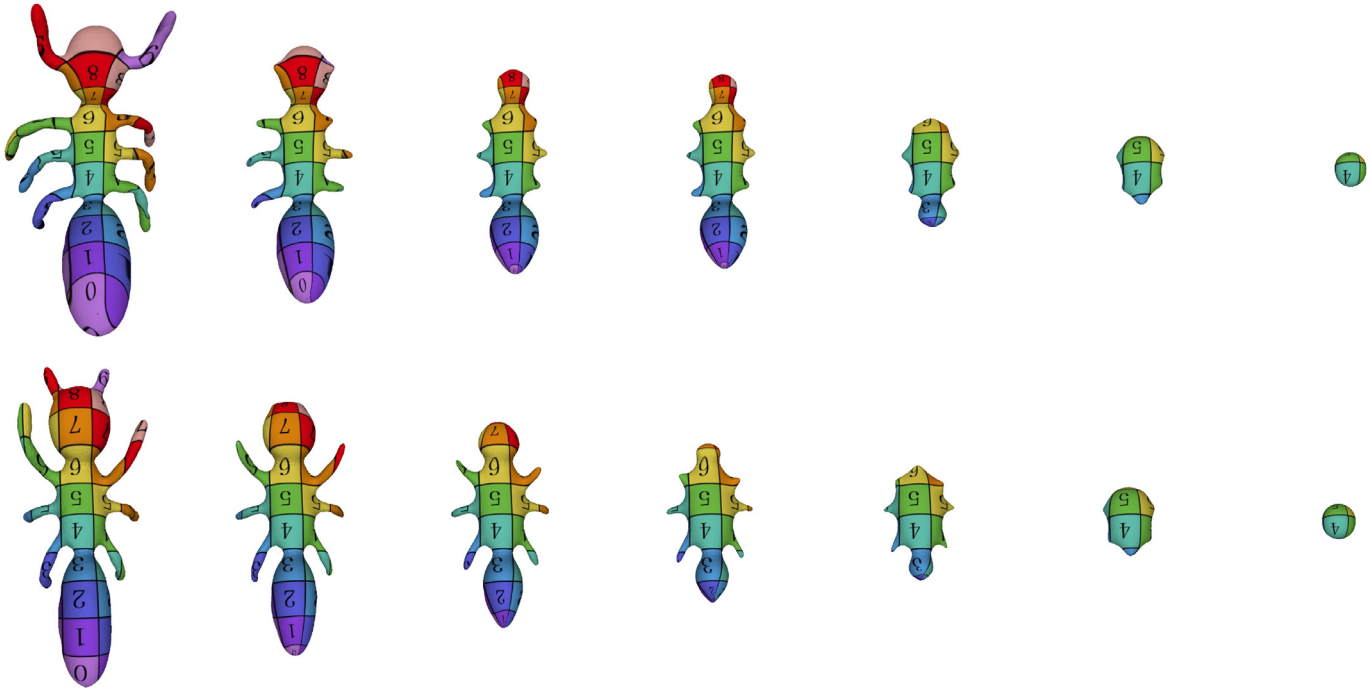

Fig. 12. Ant meshes volume mapping.

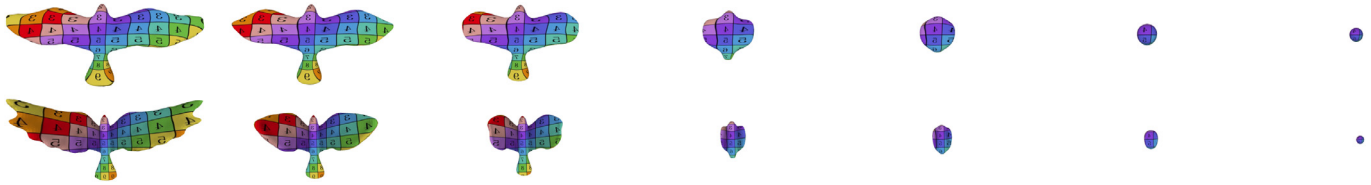

Fig. 13. Bird meshes volume mapping.

Fig. 14. Applying texture. (left) The result of our flow. (middle) Initial mesh textured. (right) Final mesh textured, zoomed in.

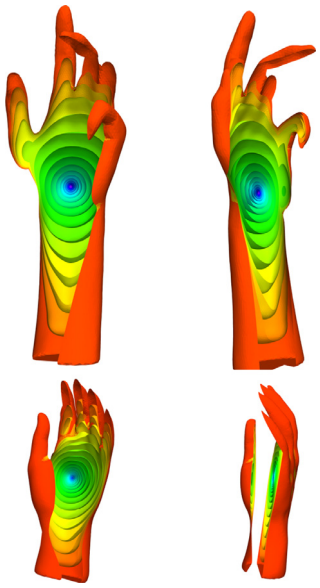

Fig. 15. Hands. In each row two different views of the same result.

singularity inside the domain. This choice was taken only for the ease of implementation. Fig. 10 demonstrates the ability of our flow to be controlled. We present 2 different meshes, each one with 3 different positions of the singular point. By changing the position of the singular point we affect the way the layers are being formed, i.e. the way the mesh converges to a sphere - the areas that will be the first and the last to move during the evolution.

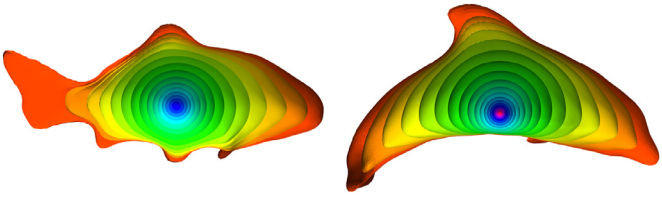

Fig. 16. Fish.

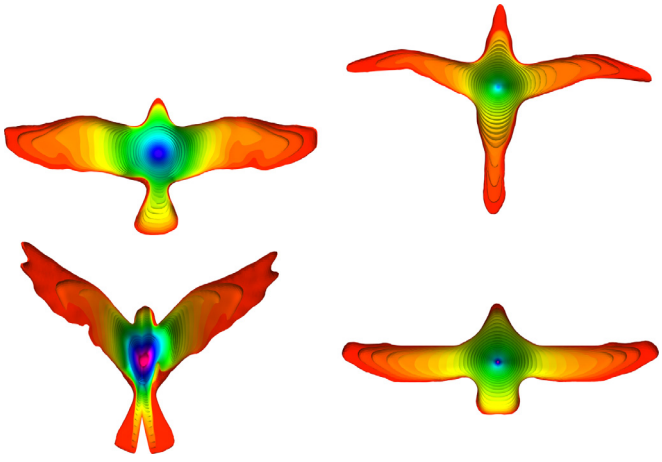

Fig. 17. Birds.

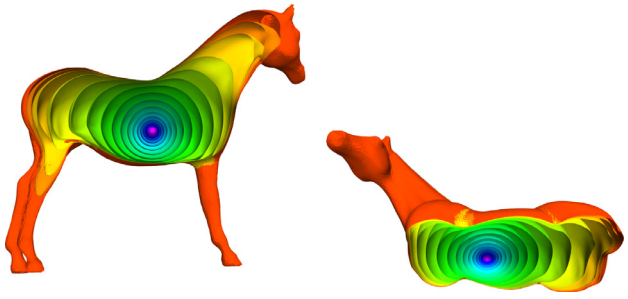

Fig. 18. Horse. (left) side view. (right) bottom view.

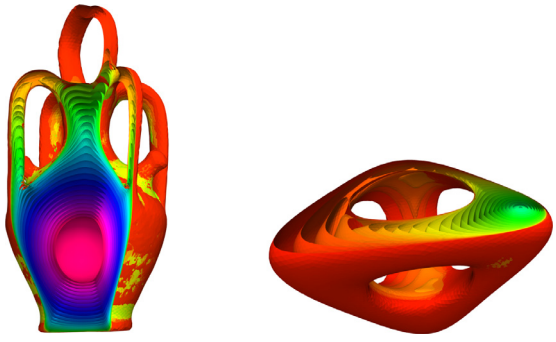

Fig. 19. High genus. (left) genus 5. (right) genus 3.

4.2. Volumetric mapping

Using our method, we can easily find a volumetric mapping, between two meshes, assuming we know a correspondence of a rather small set of point landmarks between those meshes. For building the volumetric map we utilize the method presented in [30] to get a mapping between the two initial input meshes. We then apply our method to each of the meshes until each of the meshes converges to a sphere, assuming the flow indeed

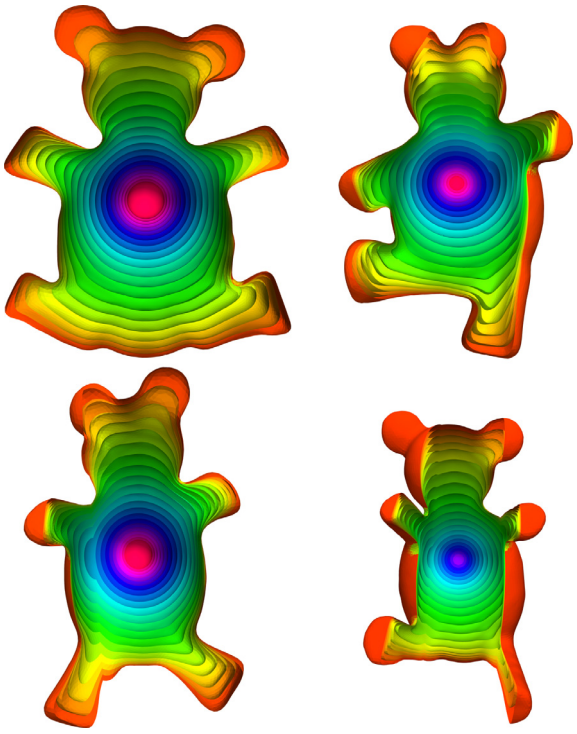

Fig. 20. Teddy meshes in different poses.

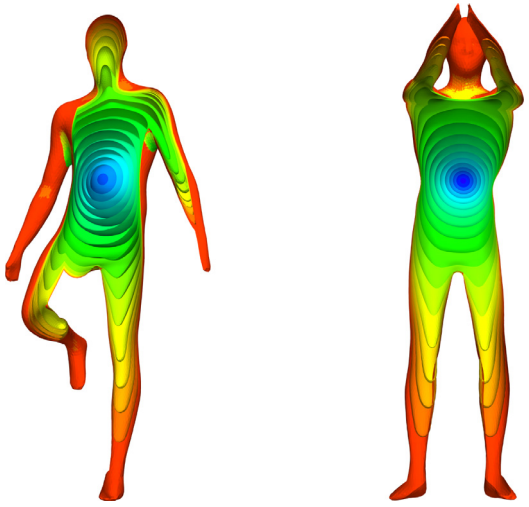

Fig. 21. Man meshes in different poses.

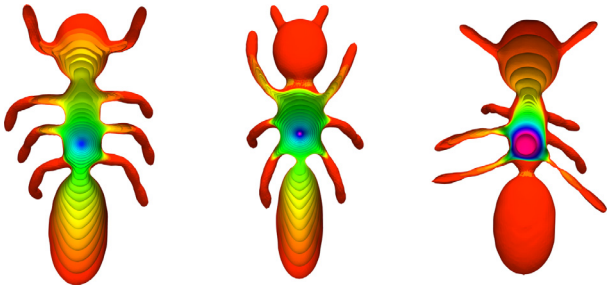

Fig. 22. Ant meshes in different poses.

converges, and having each layer in each of the evolutions with the same triangulation as the input meshes, using the method mentioned in Section 2.7. The mapping we got for the initial input meshes is then being used to map *all the matching layers* of the volumes. Furthermore, with this method we actually get a mapping between *each of the layers* of the volumes, rather than only the matching ones. Figs. 11–13 demonstrate the volumetric mapping obtained using our method.

5. Conclusions

In this paper we propose a method to generate a *generalized foliation* of a volume given its boundary surface as an input. We accomplish it by relying on a generalization of the Hele-Shaw flow with injection in three-dimensions, where the flow is being executed in the domain bounded by the Möbius inverted input boundary surface. Our *foliation* leaves are then obtained when inverting back the surfaces obtained at each simulation step. Our method produces a controllable flow that converges to a sphere. We believe that our flow could potentially be used for further mesh processing tasks, such as mesh smoothing, volumetric correspondence and volumetric texture transfer. It could also be used as a preprocess step for correspondence matching methods and for three dimensional point cloud recognition and classification algorithms. It is also interesting to investigate generalizations of the Hele-Shaw flow, such as more general singularity structures, and two-phase flows.

Acknowledgments

The authors thank Orestis Vantzos and Stefanie Elgeti for the stimulating discussions and helpful comments. M. Ben-Chen acknowledges funding from the European Research Council (ERC starting grant no. 714776 OPREP), and the Israel Science Foundation (grant no. 504/16).

Appendix A. Evolution of a sphere

An analytic solution for the model when domain is a sphere. The Laplacian in spherical coordinates (after neglecting the terms dependant of θ and ϕ due to symmetry) is,

$$\Delta\Phi = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\Phi}{\partial r}\right) \tag{A.1}$$

Eq. (2) is then,

$$\Delta\Phi = Q\delta(r) \tag{A.2}$$

The Green function for the Laplacian in spherical coordinates,

$$G(r) = -\frac{1}{4\pi r} \tag{A.3}$$

A solution is of the following form,

$$\Phi(r) = -\frac{Q}{4\pi r} + c_1, \quad c_1 \ const \tag{A.4}$$

Applying boundary condition - Eq. (7),

$$\Phi(r=R) = \sigma H \Rightarrow \sigma H = -\frac{Q}{4\pi R} + c_1 \tag{A.5}$$

$$\Rightarrow c_1 = \sigma H + \frac{Q}{4\pi R} \tag{A.6}$$

The potential is therefore,

$$\Phi(r) = \sigma H - \frac{Q}{4\pi r} + \frac{Q}{4\pi R} \tag{A.7}$$

$$\Phi(r,t) = \sigma H(t) - \frac{Q}{4\pi r} + \frac{Q}{4\pi R(t)} \tag{A.8}$$

The velocity is the normal derivative of the potential,

$$R'(t) = v = -\frac{\partial\Phi}{\partial\hat{n}}\Big|_{r=R(t)} = -\frac{\partial\Phi}{\partial r}\Big|_{r=R(t)} = -\frac{Q}{4\pi r^2}\Big|_{r=R(t)} \tag{A.9}$$

So that we get an ODE,

$$R'(t) = -\frac{Q}{4\pi R(t)^2} \tag{A.10}$$

$$\frac{dR}{dt} = -\frac{Q}{4\pi R(t)^2} \tag{A.11}$$

$$\int 4\pi R(t)^2 dR = -\int Q dt \tag{A.12}$$

$$\frac{4\pi R(t)^3}{3} = -Qt + c_2 \tag{A.13}$$

$$R(t) = \sqrt[3]{\frac{3}{4\pi}(-Qt + c_2)} \tag{A.14}$$

$$R(t=0) = R_0 \Rightarrow R_0 = \sqrt[3]{\frac{3}{4\pi}c_2} \Rightarrow c_2 = R_0^3\frac{4\pi}{3} \tag{A.15}$$

$$R(t) = \sqrt[3]{\frac{3}{4\pi}\left(-Qt + R_0^3\frac{4\pi}{3}\right)} = \sqrt[3]{R_0^3 - \frac{3Qt}{4\pi}} \tag{A.16}$$

If $Q > 0$, i.e. the flow is a suction flow, then the sphere will vanish at $t = R_0^3\frac{4\pi}{3Q}$. Our flow is an injection flow, i.e. $Q < 0$.

References

[1] Bakas I. The algebraic structure of geometric flows in two dimensions. J High Energy Phys 2005;2005(10):38, 54 pages.
[2] R Tian F. Hele-shaw problems in multidimensional spaces. J Nonlinear Sci 2000;10(2):275–90.
[3] Hamilton RS. The Ricci flow on surfaces. In: Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference in the Mathematical Sciences on Mathematics in General Relativity. University of California, Santa Cruz, California, 1986: Amer. Math. Soc.; 1988. p. 237–62.
[4] Brakke KA. The motion of a surface by its mean curvature.(MN-20). Princeton University Press; 1978.
[5] Kazhdan M, Solomon J, Ben-Chen M. Can mean-curvature flow be modified to be non-singular? Comput Graph Forum 2012;31(5):1745–54.
[6] Ye R, et al. Global existence and convergence of Yamabe flow. J Differ Geom 1994;39(1):35–50.
[7] Kuwert E, Schätzle R. Gradient flow for the willmore functional. Commun Anal Geom 2002;10(2):307–39.
[8] Yang Y-L, Guo R, Luo F, Hu S-M, Gu X. Generalized discrete Ricci flow. In: Computer Graphics Forum, 28. Wiley Online Library; 2009. p. 2005–14.
[9] Bobenko AI, Schröder P. Discrete willmore flow. In: Proceedings of the Eurographics Symposium on Geometry Processing. The Eurographics Association; 2005.
[10] Crane K, Pinkall U, Schröder P. Robust fairing via conformal curvature flow. ACM Trans Graph 2013;32(4):61:1–61:10.
[11] Gustafsson B, Vasiliev A. Conformal and potential analysis in Hele-Shaw cells. Springer Science & Business Media; 2006.
[12] Whitaker S. Flow in porous media i: a theoretical derivation of Darcy's law. Transport Porous Media 1986;1(1):3–25.
[13] Saffman PG, Taylor GI. The penetration of a fluid into a porous medium or hele-shaw cell containing a more viscous liquid. Proc R Soc Lond Ser A Math Phys Sci 1958;245(1242):312–29.
[14] Segall A, Vantzos O, Ben-Chen M. Hele-shaw flow simulation with interactive control using complex barycentric coordinates. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation. Eurographics Association; 2016. p. 85–95.
[15] Gustafsson B, Teodorescu R, Vasil'ev A. Classical and stochastic Laplacian growth. Springer; 2014.
[16] Ehresmann C, Reeb G. Sur le champs déléments de contact de dimension p completement integrable dans une variété continuement differentiable. Comptes Rendus 1944;218:955–7.
[17] Lawson Jr HB. Foliations. Bull Am Math Soc 1974;80(3):369–418.

[18] Moerdijk I, Mrcun J. Introduction to foliations and Lie groupoids, 91. Cambridge University Press; 2003.
[19] Tondeur P. Geometry of foliations. Birkhäuser Basel; 2012.
[20] Campen M, Silva CT, Zorin D. Bijective maps from simplicial foliations. ACM Trans Graph 2016;35(4):74:1–74:15.
[21] Vondenhoff E. Long-time behaviour of classical hele-shaw flows with injection near expanding balls. CASA-Report 06–19 2006.
[22] Vondenhoff E. Large time behaviour of Hele–Shaw flow with injection or suction for perturbations of balls in $\mathbb{R}^N$. IMA J Appl Math 2010;76(2):219–41.
[23] Axler S, Bourdon P, Wade R. Harmonic function theory, 137. Springer Science & Business Media; 2013.
[24] Meyer M, Desbrun M, Schröder P, Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and mathematics III. Springer; 2003. p. 35–57.
[25] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis Comput 2008;24(4):249–59.
[26] The CGAL Project. CGAL user and reference manual. 413. CGAL Editorial Board; 2018. https://doc.cgal.org/4.13/Manual/packages.html.
[27] Wang Y, Ben-Chen M, Polterovich I, Solomon J. Steklov spectral geometry for extrinsic shape analysis. ACM Trans Graph 2018;38(1):7:1–7:21.
[28] Ben-Chen M, Weber O, Gotsman C. Variational harmonic maps for space deformation. ACM Trans Graph 2009;28(3):34:1–34:11.
[29] Hu Y, Zhou Q, Gao X, Jacobson A, Zorin D, Panozzo D. Tetrahedral meshing in the wild. ACM Trans Graph 2018;37(4):60:1–60:14.
[30] Ezuz D, Solomon J, Ben-Chen M. Reversible harmonic maps between discrete surfaces. CoRR 2018;abs/1801.02453.