Advection-Driven Shrink-Wrapping of Triangulated Surfaces

Martin Čavarga

Faculty of Mathematics Physics and Informatics
Commenius University
Bratislava / Slovakia

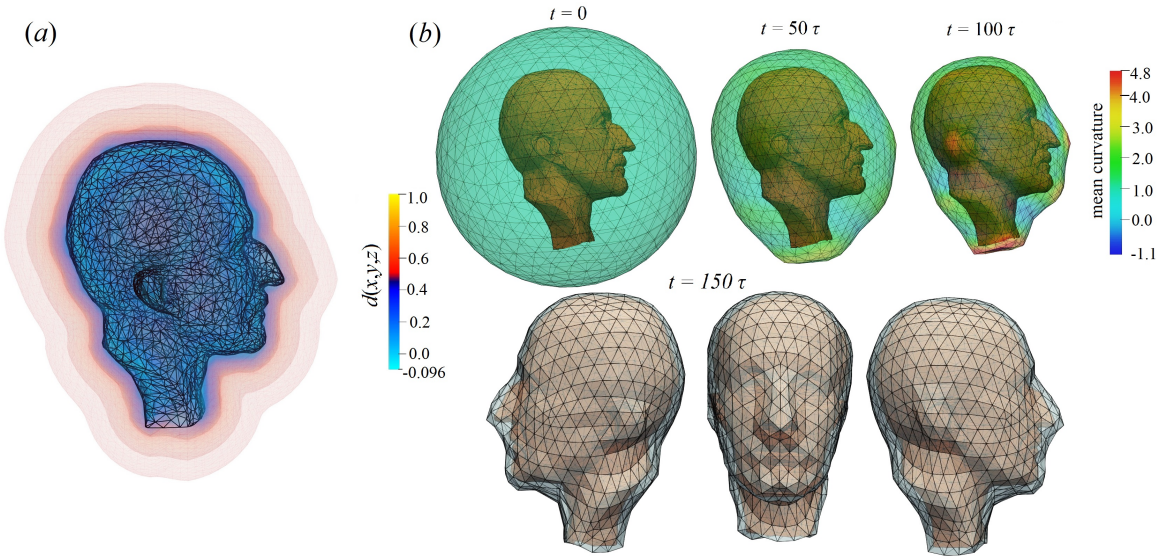

Figure 1: (a) Computed signed distance field (SDF) d of the *Bust of Max Planck* mesh with voxel resolution $152 \times 176 \times 161$, and (b) Lagrangian evolution of surface $\overline{F}$ (with initial condition $\overline{F}^0$ as a geodesic icosahedron with subdivision level 3 with $N_t = 150$ time steps of length $\tau = 0.01$, and angle-based tangential redistribution with weight constant $\omega_{\text{angle}} = 0.5$.

Abstract

Evolution methods, widely used for shape simplification and smoothing, are also used for shrink-wrapping target surfaces. Diffusion of curvature is usually the dominant driving force behind the wrapping process, along which the implicit representation of the target surface (e.g. signed distance field) takes increasing control as it is approached by the evolving surface. This process, previously used for wrapping point cloud data with triangular meshes as well as an experimental form of remeshing, can take many forms. We compare their inherent properties, versatility, and convergence rate, as well as their impact on polygon quality. The key contribution of our research is a shrink-wrapping tool for triangular to polygon target meshes with an accelerated implicit field generator as well as an analysis of mesh quality after incorporating multiple methods of tangential redistribution.

Keywords: shrink-wrapping, mean curvature flow, surface evolution, advection, fairing, signed distance field

1 Introduction

Diffusion is a well-known natural process with a multitude of manifestations in which a system evolves towards an equilibrium. Coupled with advection, this process provides means for better control of the evolving medium. If we restrict our solution to a single curve or a surface we refer to this type of evolution as *Lagrangian*. This process has been used for various applications, such as numerical construction of minimal surfaces [14], Truss structure design [10], point cloud meshing [2], and quad remeshing of triangular meshes [5]. All of the above works use an evolution model driven by diffusion of mean curvature, that is *mean curvature flow* (MCF).

With promising results for quad surface patches [5], and point-cloud reconstruction [2], Lagrangian evolution model introduced by Mikula et al. [11] can be accompanied by an advection function of an *implicit* (volumetric) *representation* of input geometry. Control over the quality of triangulation for the evolving surface is also required to avoid the formation of degenerate polygonal elements in regions with stronger MCF. Our proposed approach makes use of two tangential redistribution techniques: volume-

based [11], and angle-based [5], giving rise to an implementation of a mesh shrink-wrapping tool.

Our method utilizes the gradient of signed distance field (SDF) $d^{\pm}$ of *target geometry* Γ as the driving force for advection. To cover our computational requirements, we implement a novel technique for filling the computational domain with SDF values to mesh geometries with high polygon counts. The problem of accelerating the computation of SDFs has a variety of solutions [6] with different levels of technical difficulty. Our requirements for the discrete distance field are global regularity to avoid unnecessary gradient field discontinuities and reasonable computation time requirements for computing mesh SDF as a pre-processing step.

1.1 Contributions

This paper provides an outline of a shrink-wrapping algorithm of triangular meshes wrapping onto general polygonal meshes, with a sequence of preprocessing steps for computing the distance field. Since no real-time updates were required for meshes with high polygon counts, the procedure for SDF was implemented with a relatively low technical difficulty for implementation under the experimental requirements.

Our implementation of Lagrangian surface evolution relies on a semi-implicit finite volume scheme and is therefore subject to stability constraints. Our theoretical contribution (outlined in Section 2.4) is the analysis of numerical stability based on time step size and finite volume measures of an evolving mesh with different levels of recursive 4-to-1 edge subdivision.

The first set of experimental results involves the performance measurements of the SDF algorithm on a set of six experimental triangular meshes (see Section 4.1), including a performance comparison on three different CPUs. Lagrangian evolution is first tested against ground truth with three numerical experiments for verifying the convergence rate with respect to a *shrinking sphere* solution, as in [11]. Likewise, the model is tested on the former experimental dataset of polygonal meshes for final results.

1.2 Related Work

Besides Mikula et al. [11] forming the basis of the Lagrangian finite 2-volume (area) method on triangular, and Medľa et al. [5] extending it to quadrangular meshes, Daniel et al. [2] lay the foundation for the use of geometry shrink-wrapping using Lagrangian evolution. Likewise, the use of the Fast-Sweeping algorithm for computing SDFs of input geometries is inspired by Kósa et al. [9]. Both results were extracted from point cloud data.

In parallel, according to Sacht et al. [13], shrink wrapping of polygonal geometries can be extended to the construction of, so-called, *nested cages* which serve as a multi-resolution hierarchy of polyhedra. The earliest known use of the shrink-wrapping concept comes from Kobbelt et al. [8] where deformed surface methods for processing the geometry parametrization are used for locally adjusting the mesh quality in highly deformed regions.

1.3 Limitations

As mentioned, the combined SDF algorithm is more suited as a pre-processing step rather than for real-time updates of implicit representations. Furthermore, since the numerical stability of finite area formulation of the evolution model depends (besides other contributing factors discussed in Section 5) heavily on the time step size $\tau > 0$, additional considerations had to be made during the experimental setup. Namely, we compensated for the lack of stability by scaling the input geometry, so that the average finite area matches the time step size. In general, smaller time steps are more favorable to the numerical solver, however, since shrink-wrapped geometries with more features lead to increasing variance of evolving mesh triangle sizes, a more robust approach would require additional analysis of mesh vertex or polygon density.

2 Method

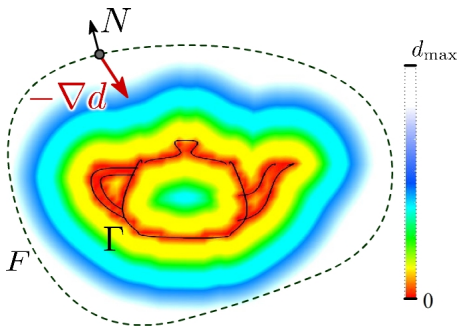

Figure 2: Distance field d to surface Γ *Utah Teapot* with resolution 120^3 and an evolving surface F driven by advection using the gradient field $-\nabla d$.

In essence, surface evolution is governed by equation:

$$\partial_t F = v_N + v_T, \tag{1}$$

where F is the immersion of an evolving 2-manifold into $\mathbb{R}^3$, and v_N with v_T are the normal and tangential velocities respectively [11]. In our model, inspired by Medľa et al. [5], evolution in the normal direction is controlled by mean curvature, and a preferred gradient field in $\mathbb{R}^3$ of the (signed) distance function $d^{\pm}$ of the original mesh. Evolution equation (1) then takes form:

$$\partial_t F = \varepsilon \Delta_{g_F} F + \eta N + v_T \ , \ F(\cdot,0) = F^0, \tag{2}$$

where F is the time-dependent evolving surface solution with initial condition F^0 (see Fig. 2), Δ_{g_F} is the Laplace-Beltrami operator[1] with respect to the current surface metric g_F, N is the outward-pointing unit normal to F, and ε, η are control functions for the two main components of

[1] a *Laplace operator* (sum of 2nd derivatives), confined to surface F.

evolution in the normal direction (summing up to velocity v_N in (1)). In particular, we use:

$$\varepsilon(d) := C_1\left(1 - \mathrm{e}^{-d^2/C_2}\right), \quad C_1, C_2 > 0, \tag{3}$$

$$\eta(d) := Cd\left((-\nabla d \cdot N) - D\sqrt{1-(\nabla d \cdot N)^2}\right), \quad C > 0,\ D \geq 0, \tag{4}$$

where d is the distance field of input surface $\Gamma \subset \mathbb{R}^3$. Contrary to Medl'a's outward-evolving formulation, in our model, there is no distinction between signed $d^{\pm}$ or unsigned d^{+} distance function to the input surface Γ. In fact, our SDF preprocessing approach (outlined in Section 2.1) automatically degenerates to computing unsigned distance field (DF) for input surfaces Γ with holes.

It should be noted that for $\varepsilon \equiv 1$ and $C = 0$ the ηN component vanishes and we are left with MCF only:

$$\partial_t F = \Delta_{g_F} F = -HN, \quad F(\cdot,0) = F^0, \tag{5}$$

where H is the mean curvature of F. The only known compact exact solution to (5) in $\mathbb{R}^n$ is the shrinking n-sphere:

$$r(t) = \sqrt{r_0^2 - 2(n-1)t}, \tag{6}$$

which will be used for numerical convergence tests in Section 4.2.

The complete model (2) with $\eta \neq 0$ is an advection-diffusion model where the diffusion component comprises of mean curvature diffusion and the advection is driven by a negative gradient vector field $-\nabla d$ of the distance function d. In fact, without full advection η with $C, D > 0$ there would be no force to drive surface evolution in regions where $\nabla d \cdot N = 0$. Hence model (2) with control constant $D = 0$ in η corresponds to a 3-dimensional obstacle problem [15] with the Laplace-Beltrami operator.

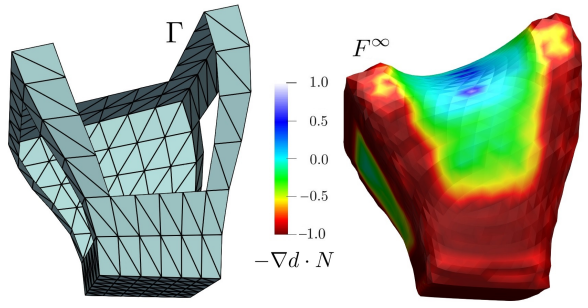

Figure 3: Shrink wrapping a deformed cube mesh with holes we refer to as *bent chair* with $D = 0$, leads to the formation of surface membranes over concave regions. The values on vertices of the triangulated limit surface F^{∞} correspond to $-\nabla d \cdot N$ where $-\nabla d$ is the negative gradient of SDF, and N the outward-pointing unit normals to F^{∞}.

Because MCF minimizes mean curvature locally, once points of F approach Γ, the MCF component vanishes with $\varepsilon = 0$. The protruding regions of input geometries Γ then serve as the limit boundary condition for the solution F as $t \to \infty$ while the surface patch with zero advection contribution η when $\nabla d \cdot N = 0$, diffuses to a minimal surface [1] (see the *bent chair* example in Fig. 3). For $D > 0$ points of F locally rotate with respect to their neighbors, since the right-hand side of (3) can actually be interpreted as incremental rotations into alignment between ∇d and N.

According to [11], for simple MCF (5), point density tends to accumulate in areas with high curvature. For this reason, we chose two mechanisms for computing tangential velocity v_T: *angle-based* and *volume-based* for the control of mesh vertex density. Although the latter notion extends to general n-dimensional volumes, for surfaces we consider 2-volumes, that is: area elements covering the evolving surface F.

2.1 SDF Mesh Preprocessing

Since distance and intersection queries are best performed on geometric primitives such as points, line segments and triangles, given an input surface $\overline{\Gamma}$ (a piecewise-linear approximation of Γ) we extract independent triangle vertex triples (v_0, v_1, v_2) as a *triangle soup* $\mathscr{T} = \{T_0, \ldots, T_{N_T}\}$. This approach extends the set of possible input geometries $\overline{\Gamma}$ to non-manifold meshes. Of course, to generate a large-enough domain G of distance field d containing $\overline{\Gamma}$ we expand the bounding box $B_{\overline{\Gamma}}$ by offset $o > 0$. The subsequent processing steps are outlined by Algorithm 1.

Algorithm 1: Computing SDF of input mesh $\overline{\Gamma}$

Data: A mesh $\overline{\Gamma}$ with extractable triangle soup $\mathscr{T}$, offset value o.
Result: A set of (signed) distance values $d_{i,j,k}$ sampled over regular grid $\overline{G}$.

1 $\mathbb{T}_{\overline{\Gamma}} \leftarrow$ generate an AABB tree from $\mathscr{T}$;
2 generate $\mathbb{O}_{\overline{\Gamma}}$ given minimum cell size $c_{\min,\overline{G}} > 0$;
3 create grid $\overline{G} \subset \mathbb{R}^3$ with dimensions of the bounding box of $\overline{\Gamma}$ and given cell resolution;
4 expand $\overline{G}$ by given offset o;
5 set exact distance values $d^{\text{exact}}_{i,j,k}$ to grid points $g_{i,j,k} \in \overline{G}$ that are centroids of octree $\mathbb{O}_{\overline{\Gamma}}$'s leaf cells;
6 set $d_{i,j,k} \leftarrow \infty$ everywhere else;
7 `fastSweep`$(\overline{G}, d_{i,j,k})$;
8 compute sign of $(\overline{G}, d_{i,j,k})$ using voxel flood fill;

The AABB (Axis-Aligned Bounding Box) tree $\mathbb{T}_{\overline{\Gamma}}$ is a binary bounding-volume hierarchy for the chosen set of geometric primitives $\mathscr{T}$ consisting of rectilinear bounding boxes $B = [b_{\min,x}, b_{\max,x}] \times [b_{\min,y}, b_{\max,y}] \times [b_{\min,z}, b_{\max,z}]$ such that leaf nodes of $\mathbb{T}_{\overline{\Gamma}}$ contain a minimal amount of geometric primitives (triangles) to iterate through. We construct and search $\mathbb{T}_{\overline{\Gamma}}$ according to the *KD-tree* algorithm in Section 5.2 of de Berg et al. [3]. The most computationally demanding step in the construction of $\mathbb{T}_{\overline{\Gamma}}$ is the search for optimal split position of box B. We use the *adaptive resampling* approach according to Hunt et al. [4].

$\mathbb{O}_{\overline{\Gamma}}$ is an octree of $\overline{\Gamma}$ constructed according to Algorithm 2 only for the purpose of sampling exact distance values $d^{\text{exact}}_{i,j,k} = d_{min}$ on a subset of grid points $g^{\text{exact}}_{i,j,k} \in \overline{G}$ corresponding to centroids of cubes $\mathscr{C}_l$ of $\mathbb{O}_{\overline{\Gamma}}$'s leaf nodes $\mathscr{O}_{N,l}$ A node of $\mathbb{O}_{\overline{\Gamma}}$ is subdivided into 8 children if and only if it intersects $\overline{\Gamma}$. These intersection queries are accelerated with $\mathbb{T}_{\overline{\Gamma}}$. The result is a *voxel outline* of a polygonal mesh in $\mathbb{R}^3$ (see Fig. 4).

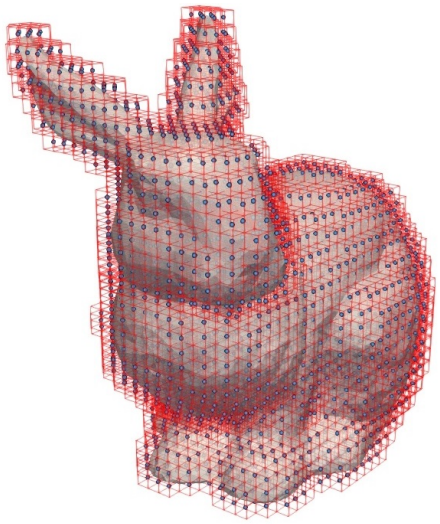

Figure 4: The *voxel outline* of a simplified *Stanford Bunny* mesh with $N_T = 5002$ triangles and voxel cell size $c_{\overline{G}} = 0.01$.

Algorithm 2: Mesh Distance-Octree

Data: An AABB tree $\mathbb{T}_{\overline{\Gamma}}$ of mesh $\overline{\Gamma}$
Result: An octree $\mathbb{O}_{\overline{\Gamma}}$ with leaves forming an outline of mesh $\overline{\Gamma}$.

1 $\mathscr{C}_0 \leftarrow$generate a bounding cube around $\overline{\Gamma}$;
2 set $\mathscr{C}_0$ as the cube of the root node $\mathscr{O}_{N,0}$;
3 **if** $\mathscr{C}_0.size() > c_{min,\overline{G}}$ **then**
4 subdivide $\mathscr{C}_0$ into 8 subcells $\mathscr{C}_k$, $k = 1,...,8$;
5 **foreach** $\mathscr{C}_k$, $k = 1,...,8$ **do**
6 **if** $\mathscr{C}_k$ *intersects mesh* $\overline{\Gamma}$ *using* $\mathbb{T}_{\overline{\Gamma}}$ **then**
7 repeat from line 3 with $\mathscr{C}_0 = \mathscr{C}_k$;
8 **else**
9 discard $\mathscr{C}_k$;
10 **end**
11 **end**
12 break if max depth is reached;
13 **else**
14 $d^2_{min} \leftarrow$ min{ squared distance of the centroid of $\mathscr{C}_0$ to $T \in \mathbb{T}_{\overline{\Gamma}}$.`getIntersectingTrianglesWith(`$\mathscr{C}_0$`)` };
15 **end**

The combination of two hierarchical structures $\mathbb{T}_{\overline{\Gamma}}$ and $\mathbb{O}_{\overline{\Gamma}}$ minimizes the computational requirements for computing exact distances by (I.) on average, $O(\log N_T)$-searching through $N_T = |\mathscr{T}|$ primitives (triangles), and (II.) uses the exact distances $d^{\text{exact}}_{i,j,k} < \infty$, and $d_{i,j,k} = \infty$ elsewhere, as an initial condition for the Fast-Sweeping algorithm [16] which sweeps $2^3 = 8$ times across points in $\overline{G}$, completing an unsigned distance field d^+ to mesh $\overline{\Gamma}$ (procedure `fastSweep` on line 7 in Algorithm 1).

Since the voxel outline of mesh $\overline{\Gamma}$ can form a *watertight boundary* of its interior[2] the subsequent optional step of a flood fill algorithm (used by Medl'a [5]) can recursively fill the approximate interior regions Int($\overline{\Gamma}$) with negative sign value leading to an approximate SDF solution $d^{\pm}$.

2.2 The Lagrangian Shrink-Wrapping Algorithm

Algorithm 3: Shrink-Wrap Remeshing of a Target Surface Γ

Data: A mesh F (preferrably of higher quality), a *target mesh* Γ, τ, N_t;
Result: Meshes F_t for each time step, and the result F_{t_s}

1 $B_{\Gamma,o} \leftarrow \Gamma$.`getExpandedBoundingBox(`$o$`);`
2 $d^{\pm}_{\Gamma} \leftarrow$`ComputeSDF(`$\Gamma, o$`);`
3 **for** $t = \tau$; $t < t_s = N_t\tau$; $t \mathrel{+}= \tau$ **do**
4 F^t.`getNormalsAndCoVolumes();`
5 **if** *do volume-based tangential redistribution* **then**
6 $h^t \leftarrow F^t$.`getCurvatures();`
7 compose and solve linear system $\mathbf{A}^{\psi,t}\psi^t = \mathbf{b}^{\psi,t}$;
8 $v^t_T \leftarrow$ F^t.`getVolumeVelocities(`ψ^t`);`
9 **end**
10 **else if** *do angle-based tangential redistribution* **then**
11 $v^t_T \leftarrow F^t$.`getAngleVelocities();`
12 **end**
13 compose and solve linear systems $\mathbf{A}^t F^{t+\tau} = \mathbf{b}^t + \tau v^t_T$;
14 `updateVertices(`$F^{t+\tau}$, $B_{\Gamma,o}$`);`
15 **end**

To declutter notation, we put $\Gamma = \overline{\Gamma}$ and $F = \overline{F}$ despite considering piece-wise linear approximations of surfaces. We start by computing the expanded bounding volume $B_{\Gamma,o}$ of input mesh Γ, and computing its signed distance field $d^{\pm}_{\Gamma}$ with expansion offset $o > 0$. Algorithm 3 then updates geometry F by first computing its *co-volumes*, that is: finite 2-volumes for the numerical method [11] (see Fig. 5), and outward-pointing unit normals N (`getNormalsAndCoVolumes`), proceeding to calculate tangential velocities v^t_T when required.

[2] such that no change of state propagating through neighboring non-diagonal voxels can reach the *interior region*. In particular if a neighboring voxel intersects Γ, it is a boundary voxel and a recursive flood may not iterate past it because it is regarded as *frozen*.

The vertex-wise contents of linear system $\mathbf{A}^t F^{t+\tau} = \mathbf{b}^t$ (without tangential velocity v_T^t) are

$$(\mathbf{A}^t F^{t+\tau})_i = \mathrm{A}_{ii}^t F_i^{t+\tau} + \sum_{p=1}^{m} \mathrm{A}_{ii_p}^t F_{i_p}^{t+\tau},$$

$$(\mathbf{b}^t)_i = F_i^t + \tau \eta_i^t N_i^t,\ i = 1,...,N_V,$$

where diagonal components A_{ii}^t corresponding to current vertex F_i^t and off-diagonal $\mathrm{A}_{ii_p}^t$ to m neighboring vertices $F_{i_p}^t, p = 1,...,m$ are determined by the finite volume formulation using the cotangent Laplace-Beltrami scheme [14]. Evolution step $t+\tau$ is finalized by verifying whether solution $F^{t+\tau}$ is bounded by $B_{\Gamma,o}$ and updating evolving mesh vertices (in `updateVertices`).

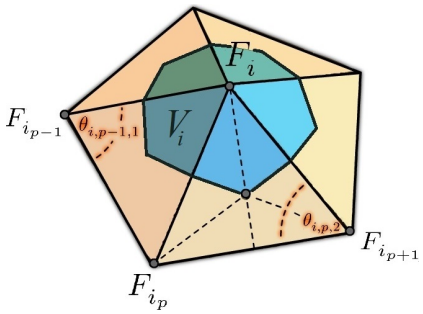

Figure 5: The definition of a *co-volume* V_i around a vertex F_i of a triangular mesh. The boundary vertices of V_i in the interiors are the barycenters of adjacent triangles. Adjacent vertices F_{i_p} contribute to the mean curvature vector approximation using cotangents of angles $\theta_{i,p-1,1}$ and $\theta_{i,p,2}$ opposing to each edge $F_i F_{i_p}$ from central vertex.

Linear systems in Algorithm 3 are solved using the stabilized bi-conjugate gradient method (BiCGStab). However, the BiCGStab method (especially with good preconditioning) is a tool which is often stronger than required for most meshes (with low vertex valence). Based on the system's diagonal dominance Mikula et al. [11] had successfully tested convergence for the SOR method as well.

2.3 Tangential Redistribution

The angle-based tangential velocity (returned by `getAngleVelocities`) is given by

$$v_{T,i} = \mathrm{proj}_T \left(\frac{\omega_{\text{angle}}}{m} \sum_{p=1}^{m} \left(1 + \frac{(F_{i_p} - F_i)}{\|F_{i_p} - F_i\|} \cdot \frac{(F_{i_{p+1}} - F_i)}{\|F_{i_{p+1}} - F_i\|} \right) ((F_{i_p} - F_i) + (F_{i_{p+1}} - F_i)) \right),$$

with weighing parameter ω_{angle}, and projection operator $\mathrm{proj}_T(v) = v - (v \cdot N)N$ with outward-pointing normal N projecting vector v to tangent plane. Inspired by Medľa et al. [5], this type of tangential redistribution homogenizes angles of adjacent polygons at central vertex F_i.

The theoretical foundations for the more complicated approach - the (asymptotically uniform) 2-volume-based redistribution are laid in [11]. The key insight from this method is that one way to homogenize co-volumes is to assume that the ratio between *volume density* and total surface area approaches a constant as $t \to \infty$. This leads to construction of a (pull-back) vector field determined by the gradient of an unknown *redistribution potential* ψ^t defined on F^t (see Fig. 6). The linear system for ψ^t comes from a finite-volume formulation of a Poisson problem which needs to be solved for each time step t. Solution ψ^t is then converted to a tangential vector field v_T by a specialized divergence formula in `getVolumeVelocities`. Curvature vectors h^t at each point are used in the Poisson problem for ψ^t.

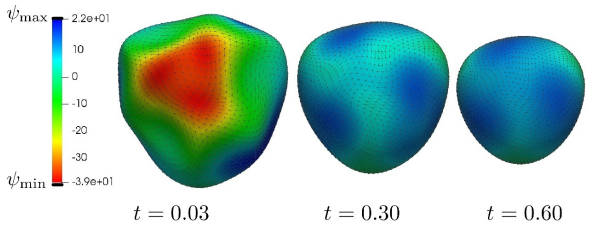

Figure 6: Redistribution potential ψ for a surface F evolving under MCF (5)

2.4 Scale Considerations for Numerical Stability

Since we use a semi-implicit formulation of a parabolic problem (2) to update mesh vertices, the stability of the numerical solution is constrained by the measures of time step τ and a spatial step h. Outside of its constraints the evolving mesh surface accumulates numerical errors that are irreversible without topological adjustments, ultimately invalidating the evolving surface solution F. For further analysis in Section 4.3, we use two key criteria for the *failure* of Lagrangian surface evolution, namely: the formation of degenerate triangles, and the explosion of mesh vertices beyond the bounding volume.

For evolving curves in $\mathbb{R}^2$, according to Section 3 in [12], the finite volume approach leads to stability constraint $\tau \approx h^2$ where h is a step in the spatial dimension of the numerical solution's domain. Hence, we put $\tau \approx \mu(V)$ where $\mu(V)$ is the co-volume measure (area). Given a characteristic dimension of bounding box $B_{\overline{\Gamma}}$ (the original box of $\overline{\Gamma}$ without offset expansion), for example, mean size, or minimum/maximum size, the factor by which a mesh needs to be scaled is:

$$\phi = \sqrt[3]{\frac{\tau}{\mu_r(V)}}, \tag{7}$$

where $\mu_r(V)$ is the mean measure of co volumes V of a geodesic icosahedron F with radius r inscribed in $B_{\overline{\Gamma}}$. Since individual areas of co-volumes V covering F are approximately equal, we put

$$\mu_r(V) = \frac{4\pi r^2}{N_V^s}, \tag{8}$$

where N_V^s is the expected vertex count of a geodesic icosahedron after s successive 4-to-1 edge subdivisions deter-

mined by a system of recurrence relations

$$N_V^s = N_V^{s-1} + N_E^{s-1}, \qquad N_E^s = 4N_E^{s-1}, \tag{9}$$

where N_E^s is the edge count of F with subdivision level s. Substituting the solution N_V^s of (9) for a given initial vertex count N_V^0, and edge count N_E^0:

$$N_V^s = \frac{1}{3}\left(N_E^0(4^s - 1) + 3N_V^0\right),$$

into (8) we obtain an estimate of $\mu_r(V)$ for any subdivision level s.

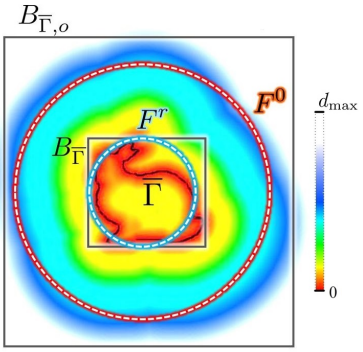

Figure 7: Under the stability assumption we consider that a geodesic icosahedron F^0 evolves into an *expected sphere* F^r with radius $r > 0$ inscribed into $B_{\overline{\Gamma}}$.

Now since the radius r of an inscribed geodesic icosahedron F depends on the size of its respective bounding box, one could extend relation (8) to any bounding box between $B_{\overline{\Gamma},o}$ and $B_{\overline{\Gamma},o}$ expanded by offset o (see Fig. 7) and interpolate between the mean co-volume measures of F^0 and F^r to compute scale factor ϕ so that the mean co-volume measure of the evolving surface F satisfies $\tau \approx \mu(V)$.

Additionally we may wish to mitigate the effects of strong angle-based tangential redistribution $v_{T,\text{angle}}$ close to target mesh $\overline{\Gamma}$ with weight function:

$$\rho(d) := A(1 - \mathrm{e}^{-d^2}),\ A > 0, \tag{10}$$

of distance d to $\overline{\Gamma}$, otherwise the influence of $v_{T,\text{angle}}$ may cause some vertices to cluster together, disrupting the assumed homogeneity of mesh vertex co-volumes.

3 Implementation

This paper's framework was implemented in C++ with architecture as shown in Fig. 8. Components Evolver and SDF Computation represent the core implementations for Lagrangian evolution and signed distance field computation respectively. The *I/O* component covers all input and output data handlings with Wavefront OBJ and Kitware™ VTK. The input geometries $\overline{\Gamma}$ were either imported as OBJ files or generated internally. Outputs in the form of VTK polydata or VTI scalar grids were then visualized using ParaView.

Furthermore, we used an additional third-party library - Poly2Tri - for $(n \geq 5)$-gon triangulation so that even geometries with higher-order polygons get converted to a triangle soup $\mathscr{T}$ used in the construction of AABB tree $\mathbb{T}_{\overline{\Gamma}}$ as the first step in Algorithm 1.

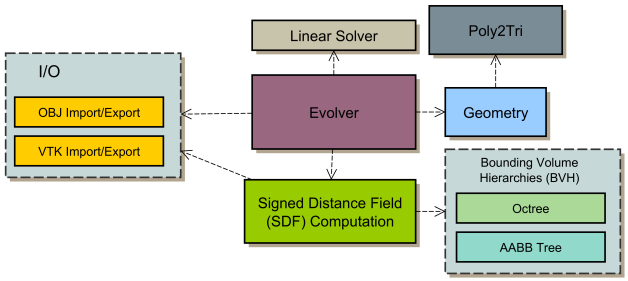

Figure 8: The architecture of our software implementation with arrows denoting dependencies between individual components.

Both the 3D geometry tools and the BiCGStab linear solver are custom-made, with mesh geometry implementation being merely indexed without additional references, requiring the use of multimap containers for mesh adjacency operations such as the computation of co-volumes and surface normals.

4 Results

4.1 Preprocessing Performance

The key preprocessing step of computing the SDF was measured via wall-clock time for a range of input *octree resolutions* corresponding to the subdivision level of the initial bounding cube $\mathscr{C}_0$. The sample volume G stemming from input geometry's bounding box $B_{\overline{\Gamma}}$ first expanded by a factor $\varphi = 1.1$ during octree construction, and later expanded by an offset $o = \max\{\beta_x, \beta_y, \beta_z\}$ where $\beta = b_{\max} - b_{\min} = (\beta_x, \beta_y, \beta_z)^\top$ is the *size (dimension) vector* of expanded bounding box $B_{\overline{\Gamma},\varphi}$. The first uniform expansion by factor φ was carried out in order to contain the mesh voxel outline, and the second to extend the distance field further into space.

We carry out the tests on six triangular meshes (see Fig. 9) with increasing grid resolution, such that for each step the time measurement is averaged over 10 runs to account for interference with other processes on our machines. We also performed the time measurements for three different CPUs, namely: AMD Ryzen© 7 3800X, Intel© i5-7300HQ, and Intel© i7-7700K.

We notice that AMD Ryzen 7 and Intel i7 perform nearly the same whereas Intel i5 lags behind by a factor of around 1.5 after the former two CPUs due to its clock speed.

For each CPU, slight improvements are achieved by using C++ *intrinsics* (configuration with *intrin* in the parentheses in Table 1) which make use of AVX registers during the demanding *adaptive resampling* step when optimal split position of bounding box is queried [4]. 256-bit vector registers, for example, allow simultaneous arithmetic and logical operations on a union of four floating point values. As before, the measured time is averaged over 10 runs.

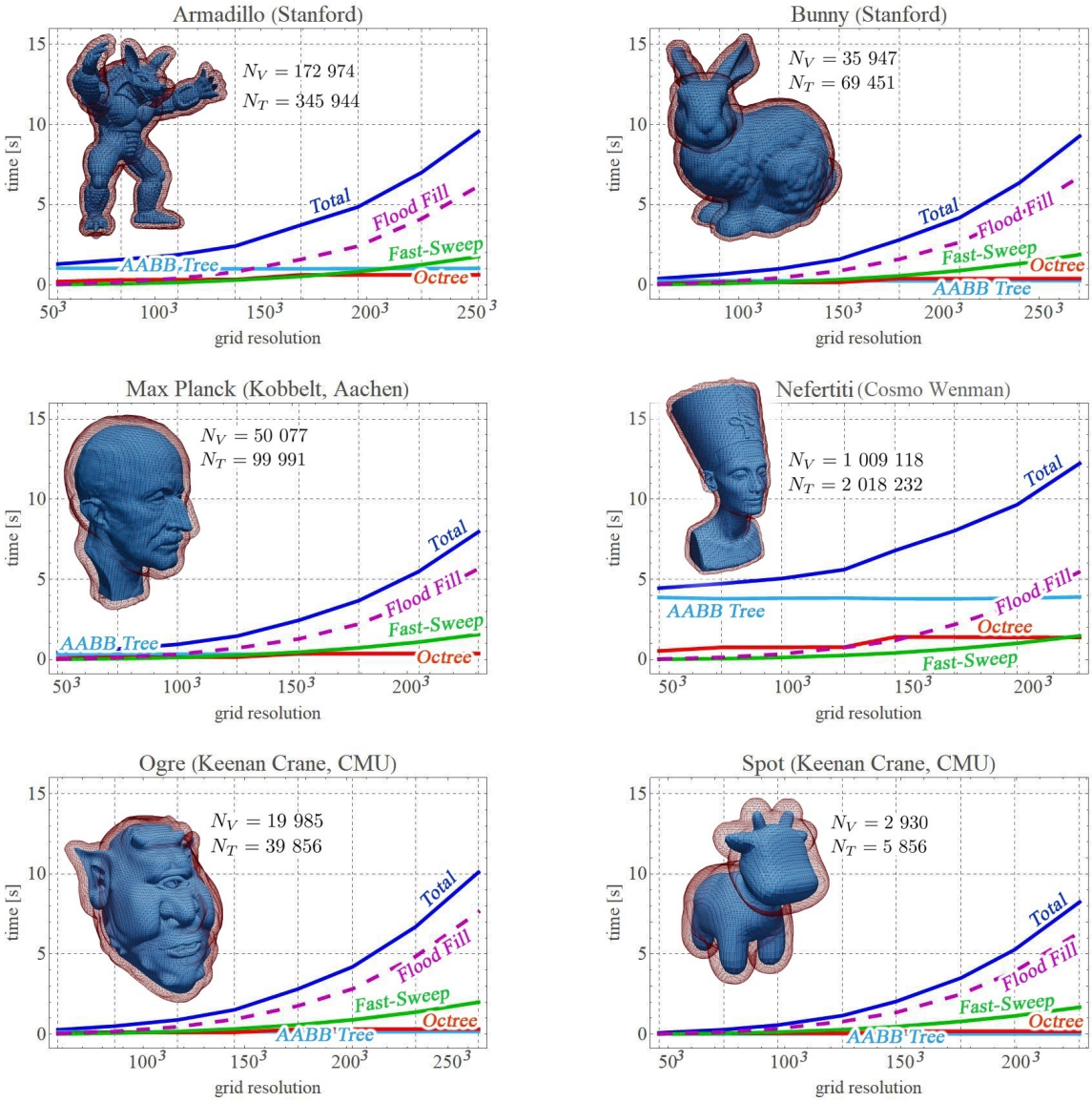

Figure 9: Wall-clock time measurements on AMD Ryzen© 7 CPU for SDF computation according to Algorithm 1, where N_V is the number of vertices, and N_T the number of triangles for each mesh.

grid resolution	90^3	120^3	180^3	240^3
AMD (intrin)	0.646	0.998	2.803	6.369
AMD (no intrin)	0.745	1.104	2.907	6.523
i7 (intrin)	0.667	1.03	2.908	6.621
i7 (no intrin)	0.766	1.197	3.058	6.755
i5 (intrin)	0.961	1.456	4.144	9.579
i5 (no intrin)	1.025	1.543	4.174	9.597

Table 1: Total time measurements (in seconds) for three CPUs of SDF computation on the *Stanford Bunny* model.

The flood fill step is the most time-consuming procedure since it needs to operate on a stack container which is generally slow with push, pop operations. Hence, the computational load increases even faster than that of the Fast Sweeping step. For geometries with millions of triangles, such as *Nefertiti*, the construction of AABB tree takes several seconds, even though this step is constant with respect to grid resolution. For grids with $\approx 100^3$ voxels, the total computation takes less than 1 second for meshes with less than 100K triangles. This is still nowhere near what some GPU-based implementations are capable of, but unless real-time updates of the distance field are required, this solution satisfies our requirements.

4.2 Numerical Experiments

Since the shrinking sphere solution (6) for $n = 3$ is directly comparable to an evolving geodesic polyhedron F with successive 4-to-1 edge subdivision, for the error

given by difference between the 2-sphere solution $r(t) = \sqrt{r_0 - 4t}$ and the *numerical sphere* $F^t = F^{k\tau}, k = 1, ..., N_t$ for all mesh vertices throughout all time steps, we use

$$\varepsilon = \sqrt{\sum_{k=1}^{N_t} \tau \left(\sum_{i=1}^{N_V} \left(\|F_i^{k\tau}\| - r(t) \right)^2 \mu_{g_F^{k\tau}}(V_i) \right)^2} \qquad (11)$$

where $\mu_{g_F^t}(V_i)$ are 2-dimensional measures of co-volumes V_i with respect to surface metric g_F^t, corresponding to mesh vertices F_i^t, and $N_t = t_s/\tau$ with stopping time $t_s = 0.06$. We put $r_0 = 1$ to perform the test on a unit sphere $\mathbb{S}^2$.

N_V	τ	N_t	ε	EOC
42	0.01	6	3.067e-03	
162	0.0025	24	8.676e-04	1.8217
642	6.25e-04	96	2.106e-04	2.0423
2562	1.5625e-04	384	5.112e-05	2.0426

Table 2: The EOC for shrinking sphere test without tangential redistribution.

N_V	τ	N_t	ε	EOC
42	0.01	6	2.845e-03	
162	0.0025	24	8.598e-04	1.7266
642	6.25e-04	96	2.105e-04	2.0305
2562	1.5625e-04	384	5.073-05	2.0528

Table 3: The EOC for shrinking sphere test with asymptotically uniform tangential redistribution with $\omega_{\text{vol}} = 1.0$.

N_V	τ	N_t	ε	EOC
42	0.01	6	3.067e-03	
162	0.0025	24	8.678e-04	1.8214
642	6.25e-04	96	2.111e-04	2.0394
2562	1.5625e-04	384	5.113-05	2.0456

Table 4: The EOC for shrinking sphere test with angle-based tangential redistribution with $\omega_{\text{angle}} = 1.0$.

Each subdivision step reduces the geodesic edge length between vertices on $\mathbb{S}^2$ to approximately half of the previous step[3], which means we can compute the *experimental order of convergence*:

$$\text{EOC} = \log_2 \left(\frac{\varepsilon_l}{\varepsilon_{l/2}} \right),$$

where ε_l and $\varepsilon_{l/2}$ are given by (11) for tessellations with geodesic edge lengths l and $l/2$. Note that since $\tau \approx h^2$ we must also reduce the length of the time step τ by a quarter. The EOC describes the rate at which the finite volume solution F^t converges to the exact $r(t)$ as we take finer tessellations of the computational domain. We choose to do so by taking halves of consecutive spatial length steps and thus take $\log_2$ of the error ratio. The results are summarized in tables (2), (3), and (4).

[3]with the approximation precision increasing after each subdivision step.

4.3 Lagrangian Shrink-Wrapping

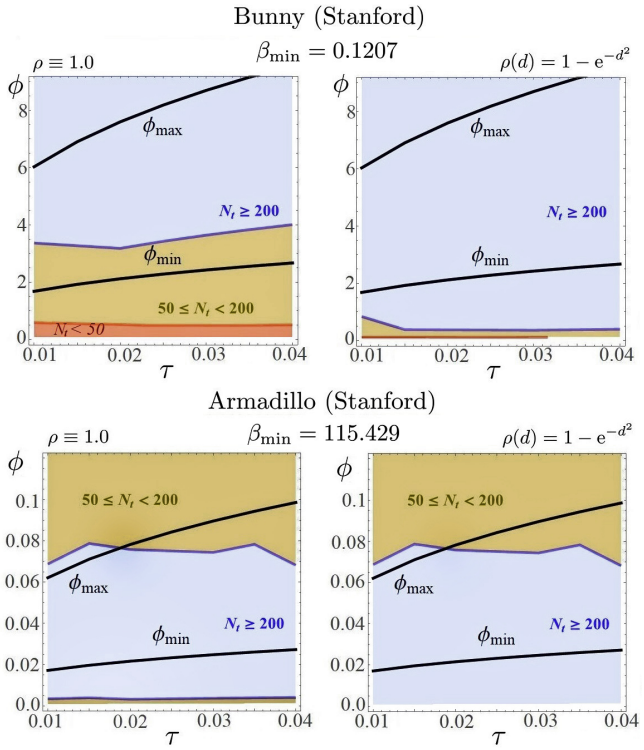

Figure 10: Regions of stability (light blue) where evolution completed $N_t \geq 200$ steps without failure. The black curves denoted by $\phi_{\min}$ and $\phi_{\max}$ are the bounds for ϕ by the starting F^0 and the *expected* surface F^r. Minimum bounding box dimensions are denoted as $\beta_{\min}$. Dark yellow and red regions correspond to evolutions that failed after $N_t \in [50, 200)$ steps and $N_t < 50$ steps respectively.

The fundamental problem in the experimental setup is the tuning of all the parameters of model (2) so that, besides other criteria, the shrink-wrapped solution F completes the process within a reasonable time while also maintaining the stability of the numerical method.

We first verify the propositions in Section 2.4 by simulating evolutions across an array of parameter values τ (time step) and ϕ (scaling factor). The results can be seen in Fig. 10 for meshes *Bunny* and *Armadillo* when the weight of angle-based tangential velocity $v_{T,\text{angle}}$ is identically one (left). As we can see, additional stabilization is achieved by using a distance-dependent weight function (10).

The stability evaluation in Fig. 10 reveals only a partial match between theoretical bounds $\phi_{\min}$, $\phi_{\max}$, and the actual stability of shrink-wrapping a specific geometry Γ with features that might lead to enlarged or stretched triangles. Other, more complicated geometries appear to have

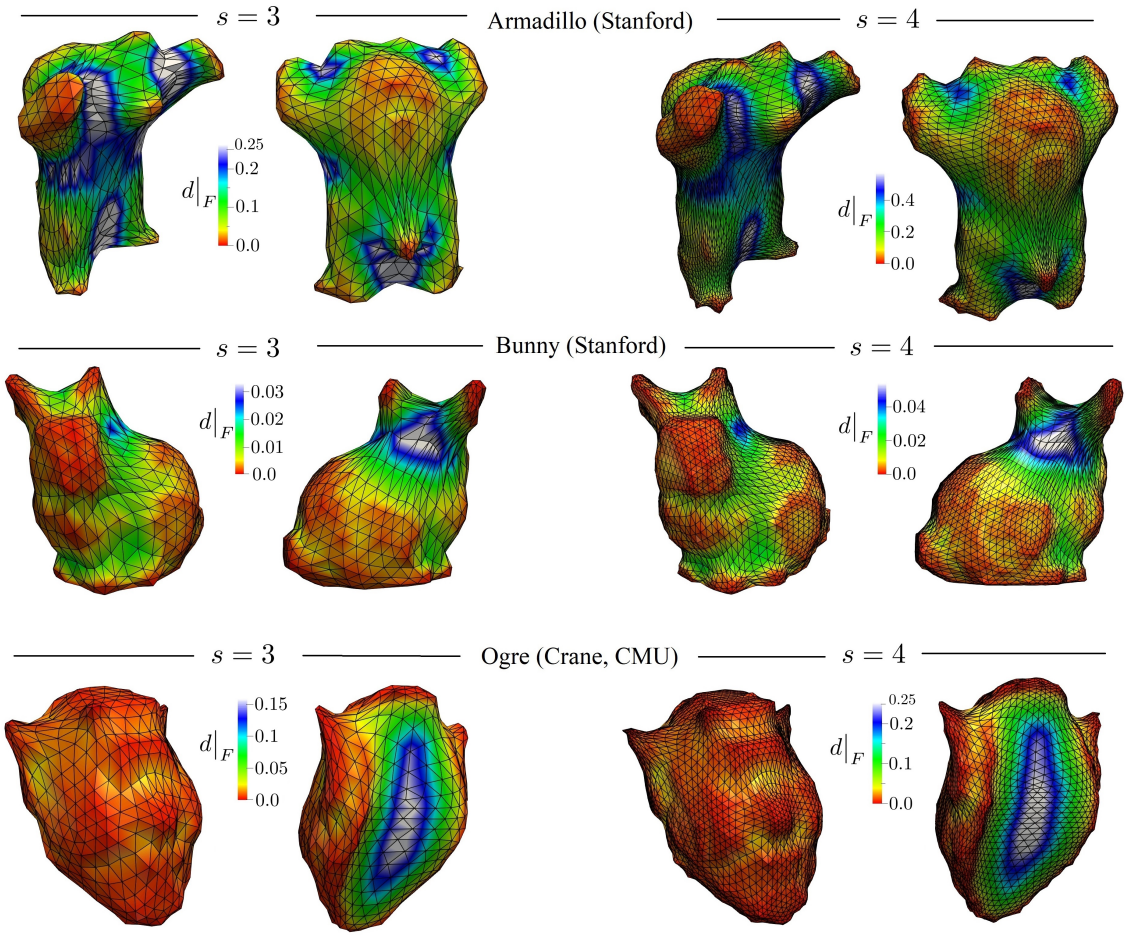

Figure 11: Results of shrink-wrapping (with $D = 0$ in η) three test meshes after $N_t = 200$ time steps of length $\tau = 0.015$ with angle-based tangential redistribution with weight $\omega_{\text{angle}} = 0.5$ for subdivision levels $s = 3$ and $s = 4$ of the evolving geodesic icosahedron surface F. The scalar values $d|_F$ are the interpolated values of distance field d on mesh vertices.

highly irregular stability regions in the τ, ϕ-plane, especially with volume-based tangential redistribution with additional stability constraints for potential ψ on F. Note that, despite evolving past the equilibrium position in concave regions around Γ, for $D > 0$ in control function η, the stretching of triangles becomes too excessive to maintain our stability assumptions. For this reason, we demonstrate the shrink-wrapping algorithm with partial advection using η with $D = 0$.

From this observation, we put $\phi = \phi_{\max}$ so that the co-volume measure is always maximized, and restrict our evolution to time step size $\tau = 0.015$. We performed Lagrangian shrink-wrapping on our test meshes the results of which can be seen in Fig. 11) for two subdivision levels $s = 3$ and $s = 4$ each with its own scaling parameter ϕ.

Besides considering *closeness* of the results to target geometries (using distance $d|_F$), we evaluated additional triangle metrics such as *minumum*, *maximum angle*, and *condition number of Jacobian* associated with the transformation from unit triangle $(0,0),(1,0),(0,1)$ in $\mathbb{R}^2$ to each triangle's planar representation [7].

5 Future Research

Although Lagrangian surface evolution models with advection fully wrap a subset of meshes, an extension of this approach to general geometries demands further inquiry. We would also like to point out that only a small sample of possible approaches to tangential redistribution have been tried. The shrink-wrapping implementation would certainly benefit from splitting and/or decimation subroutines for minimal surface regions of the solution F.

Model (2) also does not wrap holes of target geometries with higher genus. Without the need for adjusting potential self-intersections, we can start the evolution by generating a contour surface of distance function d and running model (2) after tangentially relaxing triangle vertices above the generating surface Γ.

Furthermore, with the help of connectivity adjustments on F, a more robust stability analysis is needed. Besides adjusting for vertices added during evolution, a larger portion of the parameter space of points (τ, ϕ) can be covered by an experimental design with optimized sampling.

Eventually, Algorithm 3 can be extended to quadrilateral meshes, as in [5], with additional analysis of stability for changing density of mesh vertices.

6 Conclusion

The shrink-wrapping tool using Lagrangian evolution based on mean curvature flow (MCF) comes with a robust smooth theory currently bounded by the limitations of semi-implicit finite volume method, topological properties of indexed mesh geometries, and the difficulty to tangentially redistribute mesh vertices to cluster more in regions around concave parts of input geometry Γ.

In this paper, we discussed the development of a shrink-wrapping technique for wrapping polygonal meshes. We maximized and tested the efficiency of signed distance computation of the input meshes without the requirements for manifoldness (see Section 4.1). Beyond the experimental results, we discuss and test the numerical stability of our model in sections 2.4 and 4.3.

7 Acknowledgements

The author would like to thank his master's thesis supervisor prof. Karol Mikula, and current PhD. supervisor(s) Andrej Ferko, and Robert Bohdal, for regularly taking time to discuss this project and introduce new ideas at every opportunity. Additionally, gratitude goes to Balázs Kósa for inspiration in the implementation of Fast Sweeping algorithm.

The credit for test data goes to the Stanford 3D Model Repository (Armadillo, Bunny), Keenan Crane (Ogre, Spot) and the Max Planck Institute (Max Planck), and Cosmo Wenman for the bust of Nefertiti.

Finally, the author is grateful to the anonymous reviewers for all their insights and careful evaluation.

References

[1] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. AK Peters / CRC Press, September 2010.

[2] Patrik Daniel, Matej Medl'a, Karol Mikula, and Mariana Remešíková. Reconstruction of surfaces from point clouds using a lagrangian surface evolution model. pages 589–600, 05 2015.

[3] Mark de Berg, Mark Van Kreveld, Mark Overmars, and Otfried Cheong. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[4] Warren Hunt, William R. Mark, and Gordon Stoll. Fast kd-tree construction with an adaptive error-bounded heuristic. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 81–88, 2006.

[5] Martin Huska, Matej Medl'a, Karol Mikula, and Serena Morigi. Lagrangian evolution approach to surface-patch quadrangulation. *Applications of Mathematics*, 66:1–43, 03 2021.

[6] Mark Jones, Andreas Bærentzen, and Milos Sramek. 3d distance fields: A survey of techniques and applications. *IEEE transactions on visualization and computer graphics*, 12:581–99, 08 2006.

[7] Patrick M. Knupp. Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elem. Anal. Des.*, 39(3):217–241, jan 2003.

[8] Leif Kobbelt, Jens Vorsatz, Ulf Labsik, and Hans-Peter Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18:119–130, 04 2000.

[9] Balázs Kósa, Jana Haličková-Brehovská, and Karol Mikula. New efficient numerical method for 3d point cloud surface reconstruction by using level set methods. *Proceedings of Equadiff 2017 Conference*, pages 387–396, 2017.

[10] Karol Mikula, Mariana Remešíková, and Peter Novysedlák. Truss structure design using a length-oriented surface remeshing technique. *Discrete and Continuous Dynamical Systems - Series S*, 8:933–951, 07 2015.

[11] Karol Mikula, Mariana Remešíková, Peter Sarkoci, and Daniel Ševčovič. Manifold evolution with tangential redistribution of points. *SIAM Journal on Scientific Computing*, 36:A1384–A1414, 07 2014.

[12] Karol Mikula, Daniel Ševčovič, and Martin Balažovjech. A simple, fast and stabilized flowing finite volume method for solving general curve evolution equations. *Communications in Computational Physics*, 7, 11 2008.

[13] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. Nested cages. *ACM Transactions on Graphics*, 34:1–14, 10 2015.

[14] Lukáš Tomek, Mariana Remešíková, and Karol Mikula. Computing minimal surfaces by mean curvature flow with area-oriented tangential redistribution. *Acta Mathematica Universitatis Comenianae*, 87:55–72, 02 2018.

[15] Hui Yu. A brief survey on the obstacle problem. *arXiv: Analysis of PDEs*, 2019.

[16] Hongkai Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74:603–627, 2005.