# A Variational Approach to Eulerian Geometry Processing

Patrick Mullen Alexander McKenzie Yiying Tong Mathieu Desbrun
Caltech

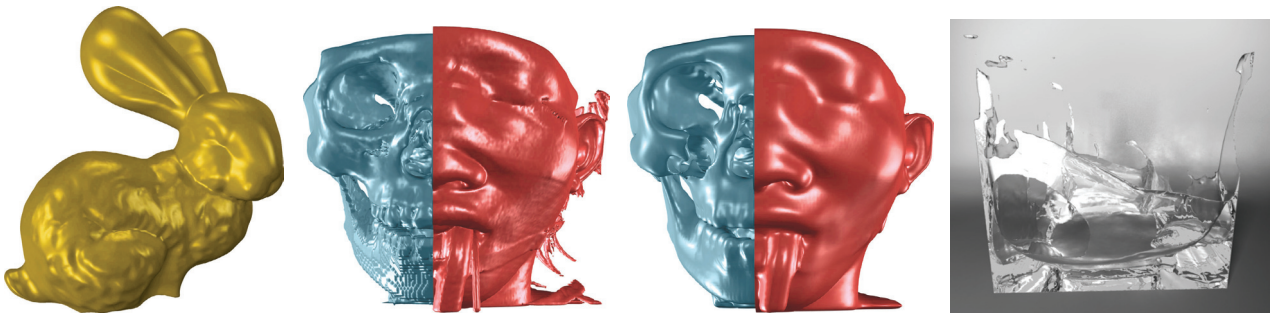


Figure 1: ***Eulerian Geometry****: our geometry processing framework offers a fully Eulerian variational approach to (left) outward and inward surface offset (here spatially varying by height), (center) simultaneous smoothing of foliations (all isosurfaces of volumetric medical data), and (right) a conservative mass advection for incompressible fluid simulation.*

## Abstract

We present a purely Eulerian framework for geometry processing of surfaces and foliations. Contrary to current Eulerian methods used in graphics, we use conservative methods and a variational interpretation, offering a unified framework for routine surface operations such as smoothing, offsetting, and animation. Computations are performed on a fixed volumetric grid without recourse to Lagrangian techniques such as triangle meshes, particles, or path tracing. At the core of our approach is the use of the Coarea Formula to express area integrals over isosurfaces as volume integrals. This enables the simultaneous processing of multiple isosurfaces, while a single interface can be treated as the special case of a dense foliation. We show that our method is a powerful alternative to conventional geometric representations in delicate cases such as the handling of high-genus surfaces, weighted offsetting, foliation smoothing of medical datasets, and incompressible fluid animation.

**Keywords:** Digital geometry processing, Offset surfaces, Normal flows, Mean curvature flow, Foliations, Fluids.

## 1 Introduction

Evolving surfaces, be it for modeling or animation purposes, is a routine task in Computer Graphics. Over the last decade, the method of choice to process geometry has consisted of a Lagrangian setup where the surface is explicitly stored as a piecewise-linear mesh, and vertices are moved so as to achieve the desired deformation [Botsch and Pauly 2006]. Great success with this approach has been reported for editing, smoothing, and parameterization, often using variational formulations [Pinkall and Polthier 1993; Grinspun et al. 2003]. Nevertheless, Lagrangian methods come with their share of difficulties, including mesh element degeneracies, self-intersections, and topology changes, all of which require delicate treatment. While some of these issues can be addressed with *point sets* [Alexa et al. 2006], the problem of continuous (fine) resampling remains, and the concern of a proper, topologically-sound surface reconstruction arises.

Consequently, Eulerian methods emerged as a great alternative to meshes in several applications [Frisken et al. 2000; Museth et al. 2002; Tasdizen et al. 2003]. One particularly successful Eulerian approach is the Level Set Method (LSM), drawing on the development of numerical solutions to the Hamilton-Jacobi equations in applied mathematics. The LSM methodology has proven very useful in vision, image processing, as well as graphics [Sethian 1999; Osher and Fedkiw 2003] since the traditional hurdles that mesh processing faces are nicely circumvented due to a parameterization-free treatment. However, other problems arise in this particular Eulerian approach. From a numerical point of view, the LSM relies on finite difference methods applied to a distance function. This specific setup has significant consequences, first and foremost being that volume loss *cannot* be prevented without using additional (most often Lagrangian) computational devices. A less obvious consequence is that the variational nature of useful flows such as mean curvature motion, which was numerically exploited and proven crucial for mesh processing [Desbrun et al. 1999], is no longer respected.

In this paper we present an alternative to current purely-Eulerian methods in graphics by describing a *conservative* (*i.e.*, mass-preserving) and *variational* treatment of basic geometry processing tasks. Based on Geometric Measure Theory, our approach even allows foliation (multiple-surface) processing, a particularly useful tool, *e.g.*, in the treatment of medical datasets (see Fig. 1 & 10).

### 1.1 Background

Computing interface motion in the Eulerian setting has recently received considerable attention in applied mathematics and computational physics (thorough reviews can be found in, for instance, [Sethian 1999; Osher and Fedkiw 2003]). These techniques have made promising contributions to geometry processing in the last few years.

**Lagrange vs. Euler** There is a clear historical preference for Lagrangian methods in surface processing, probably due to the classical parametric definition of surfaces in differential geometry. More-

over, the large number of efficient data structures available and the ease with which geometric quantities (volume, area, curvatures, etc) can be accurately evaluated have contributed to make meshes the representation of choice for surfaces. In comparison, Eulerian data structures were, until recently, limited to regular grids or restricted octrees. However, recent progress (*e.g.*, [Houston et al. 2006]) is a clear sign that fast and concise Eulerian representations can now compete with mesh-based methods—their most salient property over mesh-based approaches being the natural handling of complex topology changes without special treatment.

**Level Set Method** Over the years Eulerian approaches have even shown superiority in applications such as compression of complex surfaces [Lee et al. 2003], surface offsetting [Breen and Mauch 1999], or even surface mesh extraction from 3D scans [Curless and Levoy 1996]. In applications where high-quality surfaces need to be treated, a particular Eulerian technique called the Level Set Method has also been successfully used for surface editing [Museth et al. 2002], smoothing [Tasdizen et al. 2003], texturing [Bargteil et al. 2006], and even incompressible fluid animation [Foster and Fedkiw 2001] to mention a few. The basic idea of the LSM is to represent a surface as the zero level set of a *signed distance function*, referred to as the level set function, and to evolve this function according to a partial differential equation of motion [Osher and Sethian 1988]. This level set function is efficiently stored as discrete values on a fixed regular grid, allowing for simple Finite Difference schemes to integrate the motion.

There are, however, serious theoretical and practical issues with LSM. First, the use of a distance function brings inherent limitations: even in the continuous limit, a Lie-advected distance function is no longer a distance function. Consequently, an Eulerian discretization of this particular setup introduces an inevitable numerical drift resulting in volume loss, particularly in regions of high curvature. A multitude of remedies to this intrinsic deficiency have been proposed, sometimes as simple as refining the grid, but often at the cost of significantly higher computational complexity [Sussman and Puckett 2000; Frolkovic and Mikula 2005; Olsson and Kreiss 2005; Losasso et al. 2004; Mihalef et al. 2006; Bargteil et al. 2006]. The addition of a large amount of Lagrangian particles was introduced as a way to compensate for volume loss [Enright et al. 2002], although high-frequency perturbations of the surface can appear [Bargteil et al. 2006] (one promising approach is to store the level set on SPH-like particles [Hieber and Koumoutsakos 2005] as it benefits from a dual Lagrangian-Eulerian representation). Second, the LSM uses a traditionally cell-centered representation of vector fields, incompatible with conventional incompressible Navier-Stokes solvers that use staggered Marker-And-Cell (MAC) grids.

**Gradient Flows** Gradient flows are the linchpin of geometry processing: many now-traditional geometric tools for meshes such as mean curvature flow (MCF), shape optimization, and conformal parameterization are best expressed *and* numerically resolved through simple variational principles [Botsch and Pauly 2006; Grinspun 2006] mostly based on $L^2$-minimizations. However, Eulerian geometric methods (including LSM) rarely exploit these variational qualities to derive robust numerical schemes. In particular, the numerical implementation of MCF in LSM relies on finite differences to *directly approximate the partial differential equations*, instead of treating the underlying variational principles.

### 1.2 Our Approach

In this paper, we propose a fully-Eulerian approach to geometry processing that is numerically based on *variational principles* and designed to *preserve fundamental invariants*—two critical differences from LSM-based techniques. We show that processing *foliations* allows the treatment of single surfaces *and* multiple surfaces in a unified framework based on the Coarea Formula. Reusing existing numerical techniques as much as possible (*e.g.*, the large body of work on Eulerian advection), we go through the list of basic geometry processing tools: advection, outward and inward offset, mean curvature flow, and other gradient flows.

Our contributions include a number of distinctive features:

- We use a fully Eulerian representation for our surface(s), eliminating the need to prevent the typical degeneracies of Lagrangian mesh elements.
- Unlike LSM, volume control is facilitated by construction, allowing conservative flows (as in the case of incompressible fluid simulation) to be easily approximated.
- Gradient flows are numerically achieved through a simple variational approach based on the Coarea Formula. In particular, we perform mean curvature flow through area minimization as proposed in [Droske and Rumpf 2004]—but with no need for regularization.
- Multiple surface processing (where all the isosurfaces in a dataset are handled at once) is easily achieved.

## 2 Processing Eulerian Foliations

Before delving into the mathematical foundations of our approach, we first discuss the dedicated data structures and representations we wish to utilize. We focus particularly on finding a representation that is as simple as possible (a single value per grid cell to optimize efficiency and memory requirements), but able to capture basic geometric measures like area.

### 2.1 Discrete Setup

A fully-Eulerian setup requires special types of surface representations: we only allow ourselves to encode and process *data stored on a fixed grid*. However, the exact type of data to use is, a priori, arbitrary. Out of the various possibilities, we must rule out using the conventional LSM distance function representation: as we discussed earlier, such a setup does not appear to be a viable solution due to an inevitable loss of volume however accurate its numerical treatment is—particularly in the case of incompressible fluids (see Fig. 3 for a simple example). Unfortunately, Volume-Of-Fluid methods (VOF, where the exact ratio of occupancy of an object within each cell is stored [Puckett et al. 1997]) must also be ruled out despite their perfect volume control, due to their notorious difficulty in accurately evaluating geometric quantities such as curvatures. Note that newer variants have partially addressed this deficiency, at the price of a dramatic increase in data storage and processing [Dyadechko and Shashkov 2006]. Embracing the specificity of Eulerian grids, the Phase Field Method (PFM) by [Anderson et al. 1998] proposes instead the notion of a *smeared interface* representation, where grid cells in a finite width around the interface capture rapid but smooth transitions in density. PFM, however, presupposes the profile of the smeared interface, requiring very fine grids to obtain detailed results.

**Finite Volumes** Instead, we opt for a (cell-centered) Finite-Volume setup, where a single value per cell is stored. This setup falls in the category of *interface capturing methods*, as it defines the interface as a region of steep gradient of a characteristic-like function (as opposed to LSM-like *interface tracking methods* which treat the interface as a sharp discontinuity moving through a grid). This setup has an obvious physical interpretation: acknowledging the fact that explicitly maintaining a perfect Heaviside function of the object is impossible in the discrete Eulerian setting, we do not

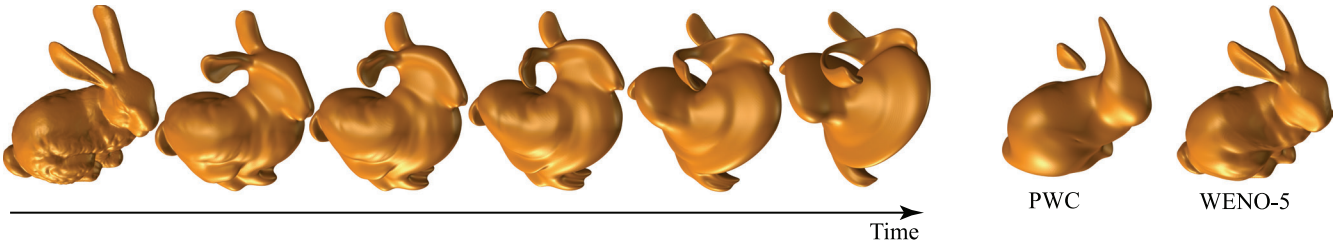



Figure 2: ***Surface Advection****: (left) the bunny is advected in a vortex flow causing severe deformation. (right) Comparison of final results obtained after reversing the vortex flow using a piecewise-constant (PWC) and WENO-5 advection scheme. Grid size:* $280^3$.

encode the exact surface, but instead store an approximate (*blurred*, in a sense) *mass density* of the object inside each cell. Thus, a cell with a value of 0 will be considered completely outside the object, a cell at 1 will be considered as completely inside, and the rest of the cells (with densities varying from 0 to 1) represent a smeared interface of the object. (Note that we will use the terms *density* and *mass density* interchangeably as our explanations will always use densities restricted to $[0,1]$.) Unlike VOF or PFM, we *do not restrict the profile* of our density function, avoiding the computational overhead incurred when a special profile needs to be maintained, as well as allowing the treatment of multiple isosurfaces with varying shape. Note finally that this density-based setup will facilitate the use of this Eulerian representation in applications such as fluid simulation.

## 2.2 Foliations

The use of smeared-out Heaviside functions to define an interface is common in phase field and level set methods (*e.g.*, when surface tension must be transferred to the surroundings). However, unlike LSM and PFM that use predefined expressions for the smearing (typically piecewise Gaussian or sine functions), we will demonstrate that there is significant benefit to keeping our approach valid for *any* function: we will be able to either accurately capture the motion of a single surface by keeping the density function sharp, or process the whole family of isosurfaces that a density function represents. Such a family of isosurfaces of a given function in $\mathbb{R}^3$ is a *foliation*, while a single isosurface is a *leaf* of this foliation (think "layers" of an onion as an analogy). At the core of our approach is the idea that one could manipulate foliations instead of single surfaces: we do not favor one isosurface over another, but rather *move them all in concert*. We show in the next section that such foliation processing can be achieved through simple volume integration, which has a natural resilience against numerical noise.

**Single Surface as a Sharp Foliation** For the treatment of single surfaces, we offer a compromise between the inherent volume loss of LSM and the artifacts found in exact volume-preserving VOF methods. Rather than trying to preserve the volume of a particular isosurface, we instead use conservative methods to exactly preserve the total mass used to represent the surface. This mass conservation leads to standard volume conservation in the limit of a sharp (unsmeared) interface, in which case our representation becomes the characteristic function of the surface. To this end, a sharpening procedure (akin to the LSM redistancing) can be employed to maintain a sharp interface and therefore give good volume control while minimizing artifacts. In particular, this exact mass preservation means simulations of moving interfaces (e.g. fluids) can be run *indefinitely* without continually accumulating volume loss.

## 2.3 Coarea Formula

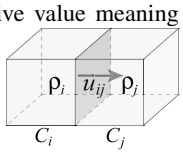
Geometric Measure Theory provides a wealth of geometric knowledge particularly relevant in graphics: its use of integration and measure theory provides generalizations of differential notions to discrete surfaces. For instance, discrete (integral) curvatures are nicely defined through Steiner's polynomial [Schröder 2006], a variational characterization of infinitesimal displacements. Of particular interest in our work is another celebrated result from Geometric Measure Theory (surprisingly absent in the graphics literature) called the Coarea Formula [Federer 1959]. When reduced to the case of 3D (Euclidean) space, this formula states that for a scalar field $\rho$ with mild continuity conditions, integrating a function over each of its isolevels in a region $\mathcal{R}$ can be done directly by a volume integral over $\mathcal{R}$ through:

$$\int_{\mathbb{R}} \int_{\rho^{-1}(c)\cap\mathcal{R}} f(\mathbf{x})\, dA\, dc = \int_{\mathcal{R}} f(\mathbf{x})\, |\nabla\rho| dV, \qquad (1)$$

where $c$ denotes an isovalue of $\rho$, $\rho^{-1}(c)$ represents the $c$-isosurface (*i.e.*, the set of 3D points such that $\rho(\mathbf{x}) = c$), and $f(.)$ is an arbitrary function of space. In other words, the term $|\nabla\rho|$ measures a local "density of isosurface area". Consequently, if we think of the foliation consisting of all $\rho^{-1}(c)$ as the representation of a "smeared interface", the Coarea Formula elucidates the relationship between the sum of area integrals and a global volume integral. We now have not only a representation, but also a mathematical formalism to process it.

## 2.4 Overview of Foliation Processing

In the remainder of this paper, we discretize a domain $\Omega$ by a regular grid $\mathcal{G}$ (extension to arbitrary grids will be discussed in Section 7). A grid cell of $\mathcal{G}$ is denoted $C_i$, and the spacing between cells is denoted $h$. A *cell average* $\frac{1}{h^3}\int_{C_i} \rho dV$ is abbreviated to $\rho_i$. We denote $F_{ij}$ the (oriented) face between cells $C_i$ and $C_j$. The *mass flux* between these cells (*i.e.*, the amount of mass per unit time crossing $F_{ij}$) is denoted $f_{ij}$ (a positive value meaning a transfer from $i$ to $j$). Finally, when using a vector field $\mathbf{u}$, we denote $u_{ij}$ its flux through the face $F_{ij}$, *i.e.*, $u_{ij} = \int_{F_{ij}} \mathbf{u}\cdot\mathbf{n}_{F_{ij}} dA$. The *step size* used for time integration is denoted $dt$.




**Foliation Evolution** In our Eulerian framework, generating a particular type of surface(s) evolution is achieved through an *update of the density field* $\rho$. As we will see next, such an update is performed via integration of a partial differential equation of the general form (different from the LSM due to its conservative and variational nature):

$$\frac{\partial\rho}{\partial t} = -\overbrace{\nabla\cdot(\rho\mathbf{u})}^{advection} + \overbrace{\frac{\partial\mathcal{L}}{\partial\rho}|\nabla\rho|}^{gradient\,flows} \qquad (2)$$

The first rhs term is an advection (*i.e.*, the surface is moved along a given vector field) corresponding to the classical mass continuity (hyperbolic partial differential) equation. The second term handles a large class of surface deformations known as gradient flows, where the gradient of an energy functional $\mathcal{L}$ is driving the surface's motion. In particular, we will show that for the *mass* functional, a

motion in the normal direction to the interface (the traditional outward or inward offset for constant propagation speed) is generated. A variational, entropy-satisfying numerical scheme will be designed to avoid artifacts like swallow tails. Another gradient flow that we will demonstrate is the mean curvature flow (when $\mathcal{L}$ is the surface area), now corresponding to a parabolic partial differential equation. We now review these two terms separately over the course of the next two sections.

## 3 Surface Advection

We first describe how our Eulerian surface(s) representation is updated to achieve advection along a given vector field $\mathbf{u}$. We follow the typical MAC set-up, *i.e.*, the vector field is given as *fluxes* on the grid cell boundaries. Note that these fluxes can be trivially computed regardless of whether the vector field is given analytically or via node values.

### 3.1 Density Advection

Since our grid stores a density value per cell, evolving the set of all isosurfaces along an external vector field simply amounts to *advecting the density*, *i.e.*, transporting mass along the velocity field. This (Lie) advection is achieved through the following equation:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho \mathbf{u}) = 0 \tag{3}$$

We can enforce this continuous equation weakly on each cell $C_i$ of our regular grid $\mathcal{G}$ through integration, as traditionally done in Finite Volume Methods, by applying the divergence theorem to the previous equation, yielding

$$\frac{\partial}{\partial t}\int_{C_i} \rho dV + \int_{\partial C_i} \left(\rho \mathbf{u}\cdot\vec{n}\right) dA = 0 \tag{4}$$

The interpretation of this last equation is particularly intuitive: mass gets moved along the vector field from one cell to the next through faces. This interpretation leads to the evolution equation of the cell averages:

$$\frac{d\rho_i}{dt} = -\frac{1}{h^3}\sum_{j\in\mathcal{N}(C_i)} f_{ij}$$

where $\mathcal{N}(C_i)$ denotes the set of cells adjacent to $C_i$, while $f_{ij}$ refers to the flux of matter between cells $C_i$ and $C_j$ induced by $u_{ij}$ as defined in Section 2.4. Because mass is only *transferred* across cells, the total mass will be preserved *by construction*. The crux of advection is thus to derive "correct" density exchanges at each cell boundary. Note the obvious link with the conventional level-set Hamilton-Jacobi equation [Sethian 1999]: for divergence-free vector fields, the two continuous partial differential equations are equivalent—only their numerical treatment differs.

### 3.2 Numerical Integration

There are a multitude of available finite-volume techniques to determine density flux through cell boundaries. The reader can find most relevant references in a recent, thorough review by Barth and Ohlberger [2004]. We remain agnostic vis-a-vis the optimal method to use. For our graphics purposes where visual impact overrides the necessity of accuracy, we opt for a dimension-by-dimension upwind interpolation. This procedure implements the conventional REA (reconstruct-evolve-average) algorithm in which the density is first locally reconstructed by a polynomial such that it fits the current cell averages, then evolved through the cell boundary, and finally averaged to deduce the quantity of mass exchange between two adjacent cells. The lowest order polynomial can often be sufficient in graphics applications, in which case we use a Godunov

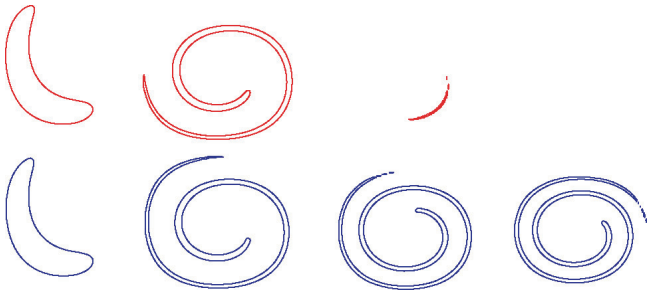


Figure 3: ***2D Comparison With LSM**: Results of advecting a circle in a vortex flow via the LSM (top) and our method (bottom), both using WENO-5. Our density-based approach continues to capture the motion long after the level set has disappeared due to volume loss during advection. Grid size:* $128^2$

piecewise-constant (PWC) approximation. This is easily computed using:

$$f_{ij} = \max(u_{ij},0)\rho_i + \min(u_{ij},0)\rho_j.$$

When higher accuracy is desirable, a WENO-5 reconstruction [Jiang and Shu 1996] is preferable as it picks an average-preserving polynomial as non-oscillatory as possible. Note that both PWC and WENO-5 are *upwinding*, *i.e.*, they have a stronger dependence on data upstream from $\mathbf{u}$—a very intuitive physical (and numerical) condition to enforce for correctly "pushing" the density along the vector field. An example of extreme surface deformation is shown in Fig. 2, where a bunny model is advected along a strong, vortex-like wind inducing large deformation. Notice that such an example done in a Lagrangian setup would require either an extremely dense triangle mesh to begin with (dense enough to handle the worst deformation), or an adaptive mesh refinement procedure to avoid artifacts. The difference that WENO-5 can make compared to PWC in quality becomes clear if the deformation is reversed: the shape of the original bunny is better preserved with a high-order advection scheme.

**Time Integration** We use a first or second-order Runge Kutta (TVD—Total Variation Diminishing) time integration [Shu and Osher 1988] depending on the targeted accuracy. Time step size may be adapted according to the maximum velocity in order to satisfy the CFL condition. Our choice of density advection technique also allows us to take larger time steps if efficiency is at stake: as detailed in [Frolkovic and Mikula 2005], the transfer-through-boundary approach can be used recursively to provide a fast, stable integration method *even for time steps larger than what the CFL stability condition imposes*—at the cost of only a small loss of accuracy.

Once an advection procedure is chosen (whichever it may be), we can proceed to define Eulerian gradient flows, as we now describe.

## 4 Gradient Flows in Eulerian Setting

Gradient flows are crucial in geometry processing, used in many design and editing tools. A case in point is the mean curvature flow (MCF) which, by following the gradient of the surface area functional, provides a geometric diffusion appropriate for denoising [Desbrun et al. 1999]. This gradient flow and its variants spawned several research topics such as conformal mapping and Laplacian editing that successfully employed the same variational setup. However, in order to define Eulerian counterparts, we first need to properly define how a geometric functional is expressed for a surface that is no longer discretized as a 2D simplicial complex, but as a smeared-out interface in space.

### 4.1 Geometric Functionals

Various geometric measures can be computed with our Eulerian density-based setup. A particularly simple (yet useful) one is the *total mass* induced by a given density $\rho$:

$$\mathcal{M}_\rho = \int_\Omega \rho \, dV, \tag{5}$$

A cell-localized mass value can similarly be defined as: $M_i = \int_{C_i} \rho \, dV = h^3 \rho_i$. Another measure we will use when dealing with a single surface is the deviation $\mathcal{D}$ of the density function from the $\frac{1}{2}$-isosurface:

$$\mathcal{D}_\rho = \int_\Omega (\rho - \frac{1}{2})^4 \, dV, \tag{6}$$

This measure will allow us to evaluate how *sharp* (*i.e.*, how close to a Heaviside function) our smeared interface is.

One useful property of the Coarea Formula is that it can be used to compute less obvious geometric measures of surfaces in the Eulerian framework. In particular, the surface area $\mathcal{A}_\rho$ of a smeared interface defined by a density field $\rho$ finds an elegant expression and interpretation. Indeed, one can take all isosurface areas into account and thus define a total Eulerian surface area (integrated throughout the foliation) as:

$$\mathcal{A}_\rho \overset{\text{def}}{=} \int_{(0,1)} \int_{\rho^{-1}(c)\cap\Omega} dA \, dc \overset{\text{Eq. (1)}}{=} \int_\Omega |\nabla\rho| \, dV \tag{7}$$

Note that, in the limit case of an infinitely fine grid where $\rho$ is exactly a characteristic function (1 for inside, 0 for outside), this integral equals the area of the (unique) interface. Therefore, this Eulerian surface area is a direct extension of the usual Lagrangian area, accommodating the volumetric nature of our spatial discretization. Using the exact same derivation, we can also define a "proportion" of total area in a cell $C_i$ as simply:

$$A_i = \int_{C_i} |\nabla\rho| \, dV.$$

### 4.2 Eulerian Gradient Flows

We approach the notion of gradient flows as a way to *evolve all isosurfaces of our density function $\rho$ in order to minimize a given functional $\mathcal{L}(\rho)$*. That is, instead of the Lagrangian formulation where a functional *on* the surface itself guides the evolution of the interface, all isodensity surfaces conspire to extremize a *volumetric* functional. This approach remains in line with our overarching methodology of treating the interface as a collection of isosurfaces, while coinciding with the Lagrangian definition in the limit of infinite resolution; it simply accounts for the reality of Eulerian discretization.

**Eulerian Norm of Variations** In our Eulerian approach, the traditional $L^2$ norm of vector fields in the Lagrangian setting must now be replaced by a special norm on variations of density $\delta\rho$: indeed, a deformation is no longer induced by a surface vector field, but by a volumetric change $\delta\rho$ of the density function. In fact, a direct application of the Coarea Formula was advocated in [Droske and Rumpf 2004] to equip the space of all possible deformation fields $\delta\rho$ with an inner product $\langle , \rangle_\rho$ through:

$$\langle \, \delta_1\rho, \, \delta_2\rho \, \rangle_\rho = \int_\Omega \delta_1\rho \, \delta_2\rho |\nabla\rho|^{-1} dV \tag{8}$$

for two variations $\delta_1\rho$ and $\delta_2\rho$ of $\rho$. We can now use this metric to define the **gradient flow of $\mathcal{L}$ with respect to $\rho$** as:

$$\frac{\partial\rho}{\partial t} = -(\frac{\partial\mathcal{L}}{\partial\rho})^\sharp = -\frac{\partial\mathcal{L}}{\partial\rho}|\nabla\rho|, \tag{9}$$

where the sharp operator $^\sharp$ uses the metric $\langle , \rangle_\rho$ to transform a differential into a vector [Abraham et al. 1988]. Note that this continuous expression is exact: no approximations have yet been made.

**Weak Form in Tangent Space** Droske and Rumpf's approach continues by defining a weak (Galerkin) formulation using test functions $\theta$ from the tangent space of $\rho$ (*i.e.*, the space in which $\partial\rho/\partial t$ lives) and enforcing that $\langle \partial\rho/\partial t, \theta\rangle_\rho = \langle -(\frac{\partial\mathcal{L}}{\partial\rho})^\sharp, \theta\rangle_\rho$. Rewritten using the density-based metric leads to the equation:

$$\int_\Omega \frac{\partial\rho/\partial t \; \theta}{|\nabla\rho|} dV = -\int_\Omega \frac{\partial\mathcal{L}}{\partial\rho}\theta dV.$$

The use of classical regularization techniques is then proposed to deal with the denominator on the lhs of this equation.

**Discrete Gradient Flows in Dual Tangent Space** We, however, prefer to avoid regularization completely. We resort instead to test functions $\theta$ from the *dual* space of $\partial\rho/\partial t$ (*i.e.*, covectors) and define our weak formulation as enforcing the equality between natural pairings (vector-covector) with all test functions, *i.e.*, $(\partial\rho/\partial t, \theta) = (-(\frac{\partial\mathcal{L}}{\partial\rho})^\sharp, \theta)$. This yields the equation

$$\int_\Omega \frac{\partial\rho}{\partial t} \, \theta dV = -\int_\Omega \frac{\partial\mathcal{L}}{\partial\rho}|\nabla\rho|\theta dV.$$

Both methods are strictly equivalent if arbitrary *continuous* test functions are used. However, we can now restrict $\theta$ to the space spanned by the piecewise-constant basis functions $\theta_i$, *i.e.*, where $\theta_i$ is defined to be 1 inside cell $C_i$ and 0 elsewhere. With this Petrov-Galerkin treatment, a gradient flow integration step is thus performed by computing:

$$\frac{\partial\rho_i}{\partial t} = -\int_{C_i} \frac{\partial\mathcal{L}}{\partial\rho}|\nabla\rho| dv \approx -\left[\frac{\partial\mathcal{L}}{\partial\rho}\right]_i |\nabla\rho|_i,$$

for each cell $C_i$, meaning that we locally increase/decrease the density according to the gradient of our functional weighted by the integrated *area* of the isosurfaces in cell $C_i$. This basic idea can now be implemented for various functionals as we describe next.

## 5 Examples of Gradient Flows

We now provide examples of basic gradient flows along with numerical details on how to implement them.

### 5.1 Surface Offsetting

We start off with the simple example of $\mathcal{L} = \rho$, that is, we wish to extremize the total change of mass. This gives $\partial\mathcal{L}/\partial\rho = 1$, in which case the gradient flow to maximize this functional becomes:

$$\frac{\partial\rho}{\partial t} = |\nabla\rho|$$

As previously mentioned, in the case of a sharp interface the total mass becomes the total volume of the surface as well. In this case the flow can be interpreted as maximizing the change in volume, which we know to be a uniform motion of the surface along its normal direction, *i.e.*, an offsetting flow. We will show that this gradient flow can indeed be used for offsetting/insetting surfaces at uniform *or* spatially-varying speeds. We will use a simple variational definition of the term $|\nabla\rho|$ as the maximum *mass gain* of a cell (induced by unit velocity) to derive its numerical approximation (the exact same argument leads to defining the inward flow that maximizes *mass loss*). For conciseness, we will denote $|\nabla\rho|_i^+$ the maximum mass gain that cell $C_i$ can receive in a time step dt, and $|\nabla\rho|_i^-$ its

maximum potential mass loss. These two values become identical for infinitely fine grids (*i.e.*, in the continuous limit), but are different in our discrete setting.

**Approximating Local Mass Gain** To obtain an accurate approximation of each maximum mass gain in the grid, we first compute the mass gain that would be incurred by each cell if the density was advected by an *axis-aligned velocity field*. This computation is easily achieved by simulating an advection step for a time step $dt$, for which the PWC or WENO-5 advection approach detailed in Section 3 is used to compute the flux on each boundary of the cell for all possible axis-aligned velocity fields. To simplify our explanations, let us switch to 2D as the extension to 3 dimensions will be straightforward. Each cell evaluates the following mass-gain values (2 values per axis, one for each direction):

$$\delta_i^{x^+} = \int_{C_i} (\nabla \cdot (\rho \begin{pmatrix} 1 \\ 0 \end{pmatrix}) dt) dV \qquad \delta_i^{x^-} = \int_{C_i} (\nabla \cdot (\rho \begin{pmatrix} -1 \\ 0 \end{pmatrix}) dt) dV$$
$$\delta_i^{y^+} = \int_{C_i} (\nabla \cdot (\rho \begin{pmatrix} 0 \\ 1 \end{pmatrix}) dt) dV \qquad \delta_i^{y^-} = \int_{C_i} (\nabla \cdot (\rho \begin{pmatrix} 0 \\ -1 \end{pmatrix}) dt) dV$$

With these values, we can now derive what the mass increase in the cell would be if the local velocity within the cell were $(u_x, u_y)$:

$$\delta\rho_i \, dt = \frac{1}{h^2} \big( \max(u_x,0)\delta_i^{x^+} + \max(-u_x,0)\delta_i^{x^-} + \max(u_y,0)\delta_i^{y^+} + \max(-u_y,0)\delta_i^{y^-} \big). \tag{10}$$

Since we are looking for the maximum mass increase for a *unit* velocity, we want to extremize the above expression subject to the constraint $|\mathbf{u}|^2 = u_x^2 + u_y^2 = 1$. This is done using a Lagrange multiplier $\lambda$, resulting in the objective function $\delta\rho_i \, dt + \lambda(u_x^2 + u_y^2 - 1)$. Setting its partial derivatives with respect to $u_x$, $u_y$, and $\lambda$ equal to 0, and substituting the solution back into Eq (10) gives the equation for the maximum mass increase for a unit velocity in the cell to be:

$$dt|\nabla\rho|_i^+ = \frac{1}{h^2}\sqrt{\begin{matrix} \max(\max(-\delta_i^{x^+},0)^2, \min(\delta_i^{x^-},0)^2) \\ + \max(\max(-\delta_i^{y^+},0)^2, \min(\delta_i^{y^-},0)^2) \end{matrix}} \tag{11}$$

Similarly, if we maximize the magnitude of mass *decrease* instead, we get:

$$dt|\nabla\rho|_i^- = \frac{1}{h^2}\sqrt{\begin{matrix} \max(\min(-\delta_i^{x^+},0)^2, \max(\delta_i^{x^-},0)^2) \\ + \max(\min(-\delta_i^{y^+},0)^2, \max(\delta_i^{y^-},0)^2) \end{matrix}} \tag{12}$$

Readers aware of the LSM machinery will notice the resemblance with the Godunov scheme for normal flows [Rouy and Tourin 1992]. Hence, our geometric derivation can draw upon applied mathematical results that proved convergence (in the sense of viscosity solutions) and entropy-satisfying property (as it avoids the formation of superfluous swallow tails) to demonstrate its validity—as well as its time step restriction. Notice that a closely related scheme to approximate $|\nabla\rho|$ is the Engquist-Osher formula [Engquist and Osher 1980]. We found that this formula can also be interpreted directly in variational geometric terms, by simply allowing the velocity to be different at each face of the cell while constraining the squared sum of all these local fluxes to be unit. This scheme is thus a viable alternative (with fairly minor visual difference) in our context as well, though its variational roots are less geometrically obvious.

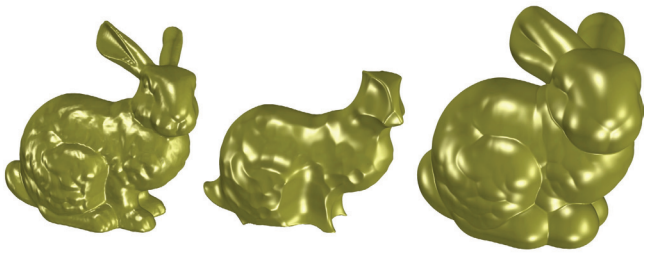
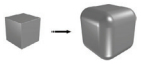


Figure 4: ***Surface Offsetting**: bunny undergoes a negative and positive offset via normal flow. Notice the sharp corners properly created in the process. Grid Size:* $350^3$.

**Implementation** Once the terms in Eqs (11) and (12) have been computed, performing a step of positive normal flow is done through:

$$\rho_i^{t+dt} = \rho_i^t + |\nabla\rho^t|_i^+ \, dt,$$



while a negative normal flow is achieved via:

$$\rho_i^{t+dt} = \rho_i^t - |\nabla\rho^t|_i^- \, dt.$$

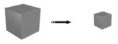


Fig. 4 provides an example of both flows on the bunny model. For a more general normal flow with a spatially-varying magnitude $\mu(\mathbf{x})$, the update becomes:

$$\rho_i^{t+dt} = \rho_i^t + \big(\max(\mu_i,0)|\nabla\rho^t|_i^+ + \min(\mu_i,0)|\nabla\rho^t|_i^-\big) \, dt, \tag{13}$$

where $\mu_i$ refers to the integral of $\mu(\mathbf{x})$ over the cell $C_i$ (see Fig. 1, left). Finally, we note that while using a WENO-5 advection is more accurate for approximating mass gain, the computationally-simpler PWC approach produces very similar visual results in practice.

## 5.2 Mean Curvature Flow

When the energy functional is the total Eulerian area (*i.e.*, $\mathcal{L} = \mathcal{A}_\rho$), the resulting gradient flow is known as the mean curvature flow. Therefore, by applying the general expression of a gradient flow, we can explicitly update the density to produce a MCF through:

$$\rho_i^{t+dt} = \rho_i^t - \frac{\partial \mathcal{A}_{\rho^t}}{\partial \rho_i}(|\nabla\rho^t|_i \, dt),$$

where $|\nabla\rho^t|_i \, dt$ is computed as detailed in Section 5.1 depending on the sign of its multiplicator. What remains to be computed is the term $\partial A_\rho / \partial \rho_i$ for a given cell $C_i$. To achieve this (again focusing on 2D for clarity), we use a first-order, centered approximation of the gradient for simplicity that yields:

$$A_i = \frac{1}{2}\sqrt{(\rho_{E(i)} - \rho_{W(i)})^2 + (\rho_{N(i)} - \rho_{S(i)})^2},$$

where $N(i), S(i), W(i), E(i)$ represent respectively the north, south, west, and east neighbor of cell $C_i$. Notice that in this low order approximation, $\rho_i$ does not appear in the above expression. Remembering that $\mathcal{A}_\rho = \sum A_i$, we can compute

$$\frac{\partial \mathcal{A}_\rho}{\partial \rho_i} = \frac{\partial A_{N(i)}}{\partial \rho_i} + \frac{\partial A_{S(i)}}{\partial \rho_i} + \frac{\partial A_{W(i)}}{\partial \rho_i} + \frac{\partial A_{E(i)}}{\partial \rho_i} \tag{14}$$

Each of these terms on the rhs is easy to compute; for example

$$\frac{\partial A_{E(i)}}{\partial \rho_i} = \frac{\rho_i - \rho_{E(E(i))}}{4A_{E(i)}} \tag{15}$$

While we explain the concept of this approach in 2D, the extension to $n$D is straightforward. Fig. 5 demonstrates how even our first-order definition of the gradient of $\rho$ provides accurate results compared to an analytical solution.

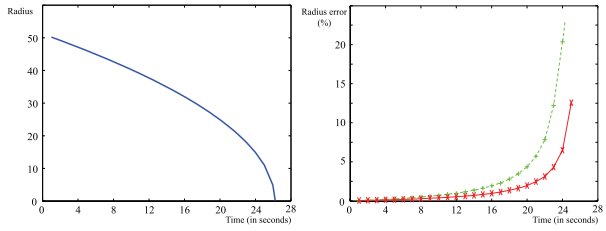



Figure 5: ***Mean Curvature Flow****: (left) a mean curvature flow of a circle results in a continuous decrease of the radius; (right) when simulated in an Eulerian setup, our approach (solid line) follows the analytical value of the radius closer than the LSM (dashed line) of equivalent order (WENO-5). Grid Size:* $128^2$

**Conservative MCF** We can now extend MCF to approximate a *conservative* mean curvature flow, a flow defined in the Lagrangian setup as minimizing surface area while preserving the volume. To implement this flow in Eulerian form, we need to first compute the global mass change $\Delta M$ induced by a given time step. This is achieved by computing

$$\Delta M = \sum \frac{\partial \mathcal{A}_{\rho^t}}{\partial \rho_i} |\nabla \rho^t|_i.$$

We then update each cell according to

$$\rho_i^{t+dt} = \rho_i^t - \left(\frac{\partial \mathcal{A}_{\rho^t}}{\partial \rho_i} - \frac{\Delta M}{\mathcal{A}_{\rho^t}}\right)|\nabla \rho^t|_i \, dt \tag{16}$$

By our definition of $\Delta M$, the total mass is exactly preserved. Note that this is analogous to the differential geometric way of writing conservative mean curvature flow, where instead of moving along the normal times $\kappa$ (mean curvature), the magnitude becomes $(\kappa - \overline{\kappa})$ where $\overline{\kappa}$ is the average mean curvature over the domain. Fig. 6 compares the conservative and non-conservative version of MCF.

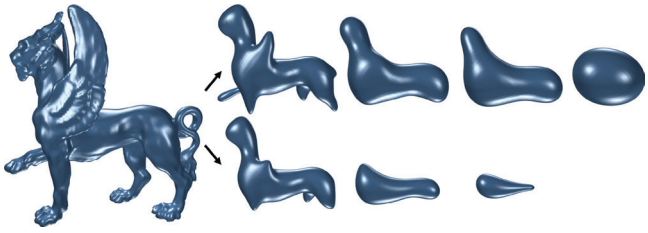


Figure 6: ***Conservative Mean Curvature Flow****: while mean curvature flow (bottom) significantly shrinks an object, a conservative version (top) restricts the flow to preserve the mass of the surface, leading to near-preservation of the volume. Grid size:* $160^3$

### 5.3 Sharpening Flow

When treating a single sharp interface the density function may smear over time, particularly with low-order schemes. As the interface gets more diffused, velocities and densities far from the interface begin to play unintended roles. Additionally, the volume of any particular isosurface can get farther from the correct value despite mass preservation. In an effort to both maintain good volume control and keep the interface reasonably sharp, we may *resharpen* the interface by redistributing the density while preserving the interface shape. A sharpening gradient flow is performed by *maximizing* the deviation function $\mathcal{D}_\rho$ defined in Eq. (6). To avoid over-sharpening (that could lead to grid aliasing), we add a limiter to the flow such that regions where the density suddenly jumps by more than a threshold $\tau$ (0.4 in all of our experiments) are left intact. Therefore, the whole sharpening phase is performed by first computing

$$w_i(\rho) = (\rho_i - .5)^3 \left(1 - \min\left(1, \frac{\max_{j \in \mathcal{N}(C_i)}(|\rho_i - \rho_j|)}{\tau}\right)\right),$$

where $\mathcal{N}(C_i)$ denotes the set of cells adjacent to $C_i$; to keep the sharpening conservative (in a manner similar to the conservative curvature flow), we then compute the total mass change $\beta$ that one step of sharpening creates and finally update each cell according to

$$\rho_i^{t+dt} = \rho_i^t + \left(w_i(\rho^t) - \frac{\beta}{\mathcal{A}_{\rho^t}}\right)|\nabla \rho^t|_i \, dt \tag{17}$$

In practice, $|\nabla \rho|_i$ need only be computed once per sharpening phase and can be reused in the few iterations needed to sharpen the band.

## 6 Applications and Implementation Details

With the above treatment of Eulerian gradient flows, our approach provides a unified framework for a wide range of geometry processing applications. We review three examples that we experimented with, and provide detailed explanations of our implementation.

### 6.1 Single Surface Processing

When focusing on a single surface (as in Fig. 2 and 4 for instance), one should maintain the sharpness of the interface such that most of the isosurfaces remain densely located in a narrow band around the surface: this strategy will drastically improve the computational efficiency by integrating the appropriate differential equations *solely inside this band*. We thus keep track of such a narrow band of cells around the interface and restrict our computations to only be performed in those band cells. When $\rho$ reaches certain thresholds (as in a negligible distance from 1, *resp.* 0) in the boundary cells of the band, we add the neighboring cells to (*resp.*, remove that cell from) the band, thus maintaining this band as it evolves in time. Any remaining small inward/outward flux across the boundary of the band is applied only to the cell in the band, while the mass change that would have been applied to the outer cell is postponed and accumulated globally. Once the magnitude of this accumulated mass reaches a threshold, it is reinjected into the system to ensure total mass preservation over time. In practice, over the course of an entire simulation the total postponed mass was generally less than half of a percent of the total mass. We stress that this narrow band and reinjection process is performed only as an optimization. In fact, while extremely valuable for many geometry processing tasks, computation on a narrow band only may not be desirable depending on the application (*e.g.*, when dealing with mixed materials or non divergence-free velocity fields).

**Reinjection** There are a few viable options on how to handle the reinjection of the postponed mass transfers. One simple way is to take a step of normal flow to increase or decrease the total mass by the necessary amount. We had better results using a second option which reinjects the mass along the advective vector field. Once the threshold is reached, we use the change from the previous advection step to add (*resp.*, remove) density only in the cells whose density increased (*resp.*, decreased), in an amount proportional to their change in that step. This has the advantage of only affecting densities in regions where it is changed regardless. While this method can cause slight inaccuracies in the total speed of the advection over long periods of time, it can be sufficient for visual purposes. A third option, preferred in our tests, is to reinject the mass during sharpening via a gradient flow, as explained next.

**Sharpening** When using a narrow band, smearing of the interface leads to a wider band and therefore a greater computational burden. For this reason the sharpening procedure explained in Section 5.3 takes on the additional role of maintaining a complexity proportional to the interface area at all times. Detecting when sharpening is necessary can be done by occasionally approximating the average width of the band: this is efficiently achieved by dividing the number

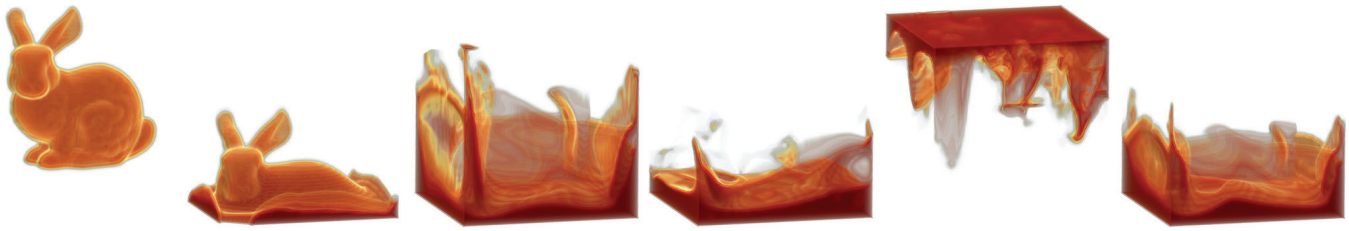


Figure 7: ***Volumetric Rendering Of Fluid****: a bunny-shaped fluid is dropped into a box, preserving the total mass even after gravity is inverted arbitrarily, causing extreme deformation. Notice how the volumetric rendering reveals intricate details captured by all the isosurfaces throughout the animation, despite a very coarse resolution (grid size:* $64^3$*).*

of cells in the band by the total surface area given in Eq. (7). If this average width exceeds some threshold, sharpening is performed as explained in Section 5.3. A few conservative gradient flow steps are taken until the band width is narrow enough. As stated above, reinjection can be performed while sharpening by subtracting the accumulated mass from $\beta$.

### 6.2 Conservative Mass Advection for Fluid Simulation

Our mass density representation accommodates incompressible free surface fluid flows seamlessly. Contrary to the level set approach, mass preservation is easily achieved through our advection scheme (Eq. 4) guaranteeing that mass is only *exchanged* through faces, and hence conserved. This property avoids the visual artifacts of volume loss traditionally present in particle-free surface capturing schemes—and without the memory and computational overhead associated with particles. In fact, having a mass density provides more intricate visualizations than the traditional single level-set visualization: Fig. 7 depicts volume renderings of a long simulation run, showing the complexity of the details captured in the density function even on a coarse grid—and mass is preserved throughout.

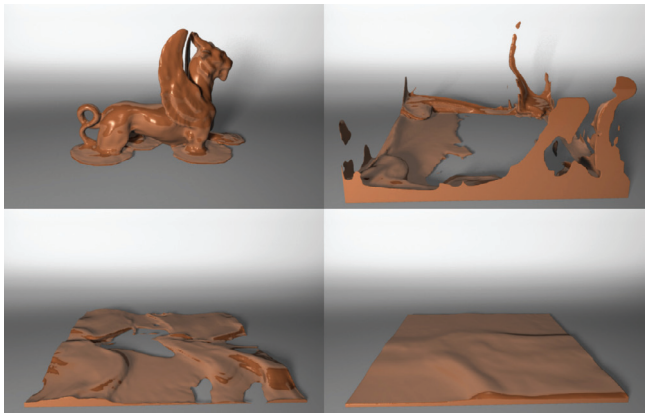


Figure 8: ***Incompressible Fluid Simulation****: the feline flows into a thin layer of liquid, showing plenty of small-scale motion while preserving volume - all in the absence of any Lagrangian artifice. Grid Size:* $128^3$ *for density,* $64^3$ *for fluid solver*

Our implementation mostly follows [Foster and Fedkiw 2001] applied in conjunction with narrow-band velocity extrapolation [Adalsteinsson and Sethian 1999], but now using our Eulerian setup instead of LSM. A cell is included in the Navier-Stokes solve if it is 'full' of fluid (*i.e.*, if $\rho_i > 1-\varepsilon$), thus allowing pressure projection to a divergence-free velocity field to be computed in the usual fashion. Note that we specify Dirichlet boundary conditions on the Poisson equation at the fluid/vacuum interface, setting the weight of the dual edge between two cells on opposite sides of the interface to $1/(\frac{1}{2}+\rho_i)$, where $i$ is the cell on the 'vacuum' side of the interface. This is a heuristic similar in spirit to the level set fluid interface alternative described by Bridson [2006], as we are ensuring that pressure is zero at the surface estimated at the distance of $\frac{1}{2}+\rho_i$ from the fluid cell. A last improvement is also added to combat the unphysical scenario of $\rho_j > 1$ (which can arise during numerical integration): we further augment the Laplacian matrix $L$ of the Poisson equation by adding $(\rho_j - 1)$ to $L_{jj}$ iff $\rho_j > 1$, or equivalently by modifying the rhs of our Poisson equation. This accentuates the pressure in the cell $j$, therefore naturally pushing excess mass into neighboring cells during the next advection time step. Notice in Fig. 8 that our density-based Eulerian formulation brings robustness to the simulation as even thin layers of fluid are treated appropriately. In the continuous setting, mass conservation and true volume preservation are synonymous under divergence freeness. While this is difficult to achieve in the discrete world, the aforementioned sharpening procedure goes far to alleviate artifacts of spatial mass diffusion, but without the visual artifacts often seen with VOF methods.

Miscible fluids can be simulated effortlessly as well, as multiple fluid densities are admissible in our representation. The total mass density per cell is directly derived as the sum of these fluid densities. To demonstrate the mixing between liquids, we use a blending of colors to indicate the types of fluid (see Fig. 9). Note that our physically based interpretation of $\rho$ as a mass density could also be amenable to high-speed *compressible* fluid models, while again maintaining mass conservation exactly.

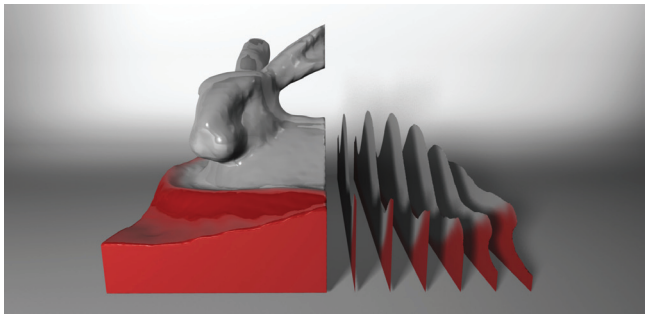


Figure 9: ***Miscible Fluids****: multiple fluid densities can be simulated in our density-based representation. Displayed are slices of two miscible fluids to demonstrate that mixing is easily achieved even on a very coarse grid (*$64^3$*).*

### 6.3 Simultaneous Mean Curvature Flows

As discussed in the introduction, mean curvature flows performed on meshes are limited: small topological defects can create degeneracies as triangles distort, requiring either iterative mesh surgery [Pinkall and Polthier 1993] or an initial topological cleanup [Wood et al. 2004]. Conversely the Eulerian framework truly shines when dealing with complex topological objects, as topology changes occur naturally without further complications. How-

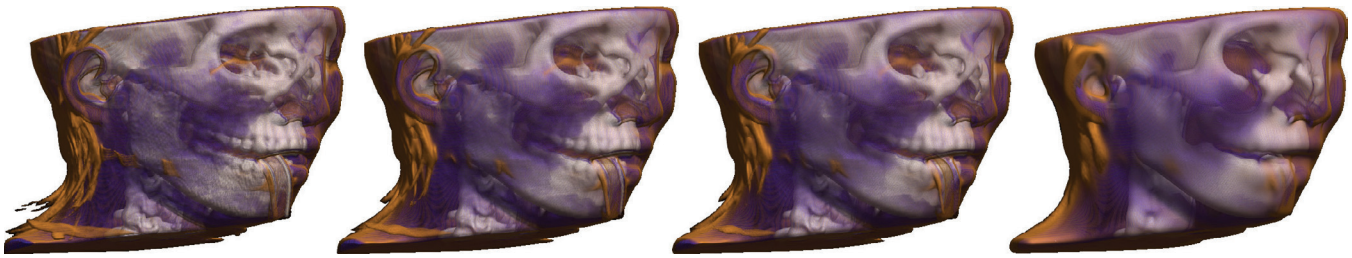


Figure 10: ***MCF Of Multiple Isosurfaces****: Foliation processing can perform a mean curvature flow of* all *isovalues concurrently. Volumetric rendering allows the visualization of several isovalues of the 3D medical scan data being simultaneously smoothed. Grid Size:* $280^3$

ever, our framework allows a nice extension: because our approach is targeting the foliation instead of a particular surface, we can apply our mean curvature flow procedure *as is* to volumetric datasets—hence performing a curvature flow of *all* isovalues collectively and concurrently. This is particularly desirable for 3D data coming from medical imaging as they represent (often noisy) density functions, *e.g.*, hydrogen nuclei density in MRI. As our representation is coherent with this density interpretation, we can smooth all the layers of the head between skin and skull simultaneously as shown in Fig. 10, while LSM or even Lagrangian methods could only process a single isosurface at a time. The speedup is therefore significant, providing a cheap, yet geometrically relevant *anisotropic filtering of 3D datasets* (comparison with a direct (isotropic) Laplacian smoothing provided in Fig. 11).

### 6.4 Discussion

In order to provide a general estimate of timings with our minimally optimized code, we mention that the advection of the bunny in Fig. 2 took 1.5 seconds per time step, the smoothing of all the isovalues of the medical data set required 30 seconds per time step, and the smoothing of the feline in Fig. 6 required 1.5 seconds per time step for non-conservative MCF and 2 seconds for conservative MCF. In general inward or outward offsetting takes times comparable to non-conservative MCF for both volumetric and single surface data. These simulations were performed on a single PC with a 2.93GHz Intel Core 2 CPU, although only the advection step was implemented to take advantage of both cores. The fluid simulations were performed on a single 1.86GHz Pentium 4 machine and took approximately 2 minutes per frame, including the fluid solve, advection, and file I/O.

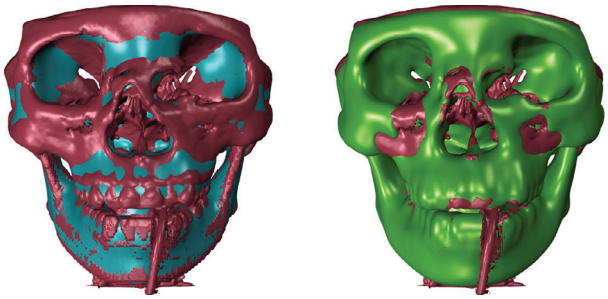


Figure 11: ***Comparison With Laplacian****: A skull (red) along with the same amount of smoothing with our mean curvature flow (teal, left) and a standard (isotropic) 3D Laplacian smoothing of the dataset (green, right). Our mean curvature behaves as expected in convex/concave regions, while Laplacian flow results in significant outward motion due to interference from neighboring isovalues. Grid Size:* $280^3$

## 7 Conclusions

We have presented a general Eulerian framework along with the necessary toolkit to perform standard geometry processing on both single surfaces and foliations through the Coarea Formula. Variational interpretations, previously proven powerful in Lagrangian settings, are used to derive simple, robust, and conservative numerical techniques for routine geometric operations such as offsetting, mean curvature flow, and animation. The applications are clearly numerous, ranging from anisotropic diffusion of 3D medical data to fluid interface simulations. Although we carefully avoid the use of Lagrangian devices to remain truly Eulerian, our framework does not prevent either (SPH) particles or semi-Lagragian path tracing to be incorporated for specific applications.

**Extensions** Although we only use explicit integration throughout our paper, implicit integration is an obvious extension deserving investigation. In particular, the mean curvature flow could benefit from the same treatment as in the Lagrangian setting where the metric was assumed constant during a time step to allow for a simple implicit update; a similar process could be applied to, *e.g.*, Eq. (15) where the denominator would be evaluated with the current value of $\rho$ while the numerator would represent $\rho$ at the next time step—resulting in a linear equation to solve for the update. Note also that while the details of our approach were restricted to regular grids, recent improvements in finite volume advection schemes [Zhang and Shu 2003] should make the extension to simplicial or cell complexes both useful and straightforward: once advection is properly defined, the exact same variational approaches we used can be extended to arbitrary grids. Advances in Eulerian data structures should also yield further computational improvements. Farther reaching future work includes investigation of higher order velocity interpolation (which may be especially useful for adaptive grids), the use of multidimensional density approximations for more accurate advection, as well as the study of higher order mean curvature (and Wilmore [Droske and Rumpf 2004]) flow approximations.

**Acknowledgments** Special thanks to Santiago V. Lombeyda for volume visualization of Figures 7 and 10. Additional thanks to Peter Schröder, Ken Museth, and anonymous reviewers for their discussions and comments. This work is supported by NSF (CAREER CCR-0133983, and ITR DMS-0453145), DOE (DE-FG02-04ER25657), and Pixar.

## References

ABRAHAM, R., MARSDEN, J., AND RATIU, T., Eds. 1988. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences Vol. 75, Springer.

ADALSTEINSSON, D., AND SETHIAN, J. 1999. The Fast Construction of Extension Velocities in Level Set Methods. *J. Computational Physics 148*, 2–22.

ALEXA, M., GROSS, M., PAULY, M., PFISTER, H., STAMMINGER, M., AND ZWICKE, M., 2006. Point-Based Computer Graphics. ACM SIGGRAPH Course.

ANDERSON, D. M., MCFADDEN, G. B., AND WHEELER, A. A. 1998. Diffuse-Interface Methods In Fluid Mechanics. *Annual Review of Fluid Mechanics 30*, 139–165.

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A Semi-Lagrangian Contouring Method for Fluid Simulation. *ACM Trans. Graph. 25*, 1, 19–38.

BARTH, T., AND OHLBERGER, M. 2004. *Finite volume methods: foundation and analysis*, vol. 1 of *Encyclopedia of Computational Mechanics, E. Stein, R. de Borst, T.J.R. Hughes (Eds.)*. John Wiley and Sons Ltd, 439–474.

BOTSCH, M., AND PAULY, M., 2006. Geometric Modeling based on Triangle Meshes. ACM SIGGRAPH Course.

BREEN, D., AND MAUCH, S. 1999. Generating Shaded Offset Surfaces with Distance, Closest-Point and Color Volumes. In *International Workshop on Volume Graphics*, 307–320.

BRIDSON, R., FEDKIW, R., AND MÜLLER-FISCHER, M. 2006. Fluid Simulation. In *ACM SIGGRAPH Course Notes*.

CURLESS, B., AND LEVOY, M. 1996. A Volumetric Method for Building Complex Models from Range Images. In *Proc. of ACM SIGGRAPH*, 303–312.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. 1999. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. *ACM SIGGRAPH*, 317–324.

DROSKE, M., AND RUMPF, M. 2004. A Level Set Formulation for Willmore Flow. *Interfaces and Free Boundaries 6*, 3, 361–378.

DYADECHKO, V., AND SHASHKOV, M., 2006. Moment-of-Fluid Interface Reconstruction. LANL Technical Report LA-UR-05-7571.

ENGQUIST, B., AND OSHER, S. 1980. One-Sided Difference Schemes and Transonic Flow. *PNAS 77*, 6, 3071–3074.

ENRIGHT, D., R. FEDKIW, J. F., AND MITCHELL, I. 2002. A Hybrid Particle Level Set Method for Improved Interface Capturing. *J. Computational Physics 183*, 83–116.

FEDERER, H. 1959. Curvature Measures. *Trans. Amer. Math. Soc. 93*, 418–491.

FOSTER, N., AND FEDKIW, R. 2001. Practical Animation of Liquids. In *ACM SIGGRAPH*, 23–30.

FRISKEN, S., PERRY, R., ROCKWOOD, A., AND JONES, T. 2000. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. In *ACM SIGGRAPH Proc.*, 249–254.

FROLKOVIC, P., AND MIKULA, K., 2005. High-resolution Flux-based Level Set Method. Preprint 2005-12. Department of Mathematics and Descriptive Geometry, Slovak University of Technology, Bratislava.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete Shells. In *Symposium on Computer Animation*, 62–67.

GRINSPUN, E., 2006. Discrete Differential Geometry: an applied introduction. ACM SIGGRAPH Course.

HIEBER, S. E., AND KOUMOUTSAKOS, P. 2005. A Lagrangian particle level set method. *J. Comput. Phys. 210*, 1, 342–367.

HOUSTON, B., NIELSEN, M., BATTY, C., NILSSON, O., AND MUSETH, K. 2006. Hierarchical RLE Level Set: A Compact and Versatile Deformable Surface Representation. *ACM Trans. on Graphics 25*, 1, 151–175.

JIANG, G.-S., AND SHU, C.-W. 1996. Efficient Implementation of Weighted ENO Schemes. *J. Comp. Phys. 126*, 1, 202–228.

LEE, H., DESBRUN, M., AND SCHRÖDER, P. 2003. Progressive Encoding of Complex Isosurfaces. In *Proc. of ACM SIGGRAPH*, 471–476.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph. 23*, 3 (Aug.), 457–462.

MIHALEF, V., UNLUSU, B., METAXAS, D., SUSSMAN, M., AND HUSSAINI, M. 2006. Physics-based Boiling Simulation. In *EG/ACM SIGGRAPH Symp. on Comput. Anim.*, 317–324.

MUSETH, K., BREEN, D., WHITAKER, R., AND BARR, A. 2002. Level Set Surface Editing Operators. *ACM Trans. on Graphics 21*, 3, 330–338.

OLSSON, E., AND KREISS, G. 2005. A Conservative Level Set Method for Two Phase Flow. *J. Comput. Phys. 210*, 1, 225–246.

OSHER, S., AND FEDKIW, R. 2003. *Level Set Methods and Dynamic Implicit Surfaces*, vol. 153 of *Applied Mathematical Sciences*. Springer-Verlag, New York.

OSHER, S., AND SETHIAN, J. E. 1988. Fronts Propagating with Curvature-dependent Speed: Algorithms based on the Hamilton-Jacobi Formulation. *Journal of Computational Physics 79*, 12–49.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics 2*, 1, 15–36.

PUCKETT, E. G., ALMGREN, A. S., BELL, J. B., MARCUS, D. L., AND RIDER, W. J. 1997. A High-order Projection Method for Tracking Fluid Interfaces in Variable Density Incompressible Flows. *J. Comput. Phys. 130*, 2, 269–282.

ROUY, E., AND TOURIN, A. 1992. A Viscosity Solutions Approach to Shape-From-Shading. *SIAM J. on Num.l Analysis 29*, 3, 867–884.

SCHRÖDER, P. 2006. *What Can We Measure*. Discrete Differential Geometry, E. Grinspun, P. Schröder, M. Desbrun (Eds.). ACM SIGGRAPH Course Lecture Notes, 8–12.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods*, 2nd ed., vol. 3 of *Monographs on Appl. Comput. Math.* Cambridge University Press, Cambridge.

SHU, C., AND OSHER, S. 1988. Efficient Implementation of Essentially non-Oscillatory Shock Capturing Schemes. *J. Sci. Comput. 77*, 439–471.

SUSSMAN, M., AND PUCKETT, E. G. 2000. A Coupled Level Set and Volume-Of-Fluid method for Computing 3D and Axisymmetric Incompressible Two-phase Flows. *J. Comput. Phys. 162*, 2, 301–337.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2003. Geometric Surface Processing via Normal Maps. *ACM Trans. on Graphics 2*, 4, 1012–1033.

WOOD, Z., HOPPE, H., DESBRUN, M., AND SCHRÖDER, P. 2004. Removing excess topology from isosurfaces. *ACM Trans. Graph. 23*, 2, 190–208.

ZHANG, Y., AND SHU, C. 2003. High-Order WENO Schemes for Hamilton-Jacobi Equations on Triangular Meshes. *J. Sci. Comput. 24*, 1005–1030.