Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

Near-Isometric Level Set Tracking

Michael Tao[1] Justin Solomon[2] Adrian Butscher[3]

[1]University of Toronto
[2]Massachusetts Institute of Technology
[3]Autodesk Research

Abstract

Implicit representations of geometry have found applications in shape modeling, simulation, and other graphics pipelines. These representations, however, do not provide information about the paths of individual points as shapes move and undergo deformation. For this reason, we reconsider the problem of tracking points on level set surfaces, with the goal of designing an algorithm that — unlike previous work — can recover rotational motion and nearly isometric deformation. We track points on level sets of a time-varying function using approximate Killing vector fields (AKVFs), the velocity fields of near-isometric motions. To this end, we provide suitable theoretical and discrete constructions for computing AKVFs in a narrow band surrounding an animated level set surface. Furthermore, we propose time integrators well-suited to integrating AKVFs in time to track points. We demonstrate the theoretical and practical advantages of our proposed algorithms on synthetic and practical tasks.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Implicit representations of geometry are critical in many computer graphics pipelines. Whereas animating and editing meshed geometry can require complex topological manipulations, resampling, and other expensive operations, implicit shapes undergo these changes smoothly while maintaining a fixed resolution by moving over a background grid of ambient space. This Eulerian perspective is popular in simulating dynamic physical materials like fluids [Bri08] and in modeling shapes out of simple primitives [Sha02, PT92] or "blob"-like building blocks [Bli82, Max83].

Animated implicit shapes are represented as level sets of a time-varying function. A key difference between this representation and an animated mesh, however, is that notions of *temporal coherence* are much weaker. Specifically, animated implicit surfaces do not encode any information about the path of a single surface point over time, needed for transporting texture coordinates, labels, and other fine-grained detail. For example, the implicit representations of a stationary sphere and a rotating sphere are identical.

To address this difference in expressive power, in this paper we reconsider the problem of recovering the velocity of an implicit surface, first introduced in [SS11]. While preserving the advantages of computing velocities using information only from a small window of frames, we propose a new model that recovers not only translational motion but also rotations and elastic deformation.

Our main technical tool is the use of approximate Killing vector fields (AKVFs), introduced to the graphics community for geometry processing in the sequence of papers [BCBSG10, SBCBG11b, SBCBG11a]. Killing vector fields are by definition the velocities of isometric motions, making them a reasonable space of velocity fields for moving and deforming implicit bodies. Since the optimization for AKVFs attempts to recover rigid motion as much as possible, by definition our method recovers rigid motions and has reasonable behavior as deformations become non-rigid.

After adapting the mathematics of AKVFs to implicit geometry, we show how to recover these fields from animated implicit sequences via one sparse linear solve per frame, solving a discretization of an elliptic PDE on a narrow band of the implicit function around the surface. No meshing is needed in our pipeline. We couple this computation with an exponential integrator for tracking individual particles over time, designed to recover both rotational and translational rigid motion exactly. Furthermore, a clear distinction of our work is that our algorithm is designed for discrete data since the PDE formulation improves its resilience to noise.

We demonstrate AKVF-based implicit velocity computation on benchmark tests to compare with previous work on the same problem. Our method is able to recover isometric motions as well as localized and global distortions of animated implicit surfaces. Additionally, we show how this new model can be applied to problems including procedurally-defined and geometrically-driven implicit surfaces represented solely by discrete level set function data. Examples include tracking animated level sets built from primitives and from mean curvature flow. Our method is thus applicable in cases where it is difficult or impossible to represent the moving

surface in an explicit fashion. Finally, when the surface motions result in topological changes we propose behaviors that we deem as appropriate despite the inherent ambiguity of this scenario.

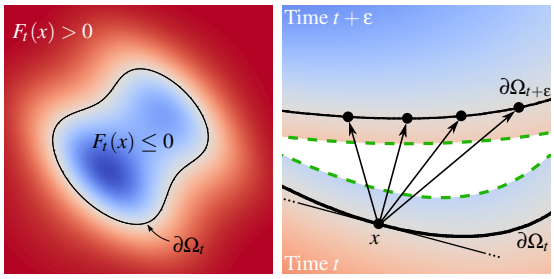
2. Related Work

Level set tracking. Tracking an interface represented as the zero-level set of a time-varying function often requires knowledge of the velocity of the interface. Often the velocity field can be derived as part of the simulation pipeline: for example, in a fluid simulation, the Navier-Stokes equations are solved for the fluid velocity and pressure [EFFM02]; or in a geometric flow problem (such as mean curvature flow), the velocity can be computed from the geometry of the interface itself [OS88]. With a known velocity field, one can compute correspondences between level sets at different times, for example [PHKF03] which advects the backward correspondences along with the level sets. A different approach, making indirect use of velocities, is to seed the implicit surface with triangulated particles and use normal projection and tangential relaxation to track the particles over time while maintaining triangle quality, e.g. [BN07]. These ideas have been applied with good success in skinning, in which an implicit representation of an animated character's body drives the movement of a textured mesh, e.g. [VGB*14].

In many cases, however, a canonically-defined interface velocity is lacking. A common practice is to use the normal velocity of the surface, e.g. in [WH94]. This is sufficient for tracking the interface, but not for tracking particles situated on the interface since normal motion alone can cause the distribution of particles to stray significantly from uniform. An *ad hoc* tangential component of the velocity is needed to ensure proper point correspondences between level sets at different times.

The question of determining the tangential component of the velocity of a moving family of level sets was first posed in [SS11]. In this work, the authors propose a local condition that can be solved for the tangential component independently at each vertex of the background grid. They assert that the normal vector of the level sets should remain constant throughout the motion. Though simple and accurate for translational motions, this condition fails to capture rotational motions. The papers [FMM13, MUM14] attempt to rectify this with another local condition based on more geometric information. Respectively, the authors propose that the curvature of the level sets should remain constant over time, and that rigid motions are solved for exactly by considering the motion of the center of mass and moment of inertia of the zero-level set. Though these methods are sufficient to capture rigid motions in certain cases, their drawbacks are that these methods may yield poor results under non-rigid motion and it is not clear how the curvatures or moments of inertia from one frame to the next should be related to each other. Furthermore, [FMM13] depends on high order derivatives of the level set function and thus is highly susceptible to noise.

Finally, the method of [DYT05] computes a function by solving the Laplace equation on the three-dimensional surface obtained by sweeping the level sets out in space-time and integrates its gradient flow to find trajectories connecting points on level sets at different times. This interesting method achieves good results but does not guarantee that rigid motions are correctly captured over time.



(a) Notation (b) Tangential ambiguity

Figure 1: Mathematical formulation. (a) Ω_t is the set of x with $F_t(x) \leq 0$, with boundary $\partial\Omega_t$; here, values of F_t are colored from blue (negative) to red (positive). (b) The constraint that $x \in \partial\Omega_t$ stays on the $F_t(x) = 0$ level set is not sufficient to determine its position as time progresses, since it can slide tangentially.

Approximate Killing vector fields (AKVFs). AKVFs were introduced in [BCBSG10] for finding continuous one-parameter families of near-isometries of a discrete surface represented as a triangle mesh. In [SBCBG11b], the spectrum of the Killing operator was used to segment the mesh into primitives with similar sets of local near-isometries. [ABCCO13, AOCBC15] find AKVFs using the methodology of functional maps [OBCS*12]. Closer in spirit to our application is [SBCBG11a], where the authors find large deformations of planar shapes by integrating two-dimensional AKVFs satisfying user-prescribed constraints (deformations of key vertices).

3. Mathematical Formulation

3.1. Moving Objects as Families of Level Sets

Consider a compact body moving either rigidly or non-rigidly through space in a unit time interval. Let $\Omega_t \subseteq \mathbb{R}^3$ be the body at time t and let $\partial\Omega_t$ be its boundary. If we use the *level set representation* to describe Ω_t, then we have a family of functions $F_t : \mathbb{R}^3 \to \mathbb{R}$ for which Ω_t is the sub-level set $\{x \in \mathbb{R}^3 : F_t(x) \leq 0\}$ and $\partial\Omega_t$ is the level set $\{x \in \mathbb{R}^3 : F_t(x) = 0\}$. Figure 1a illustrates our notation.

We emphasize that the choice of level set function is not unique: e.g. for any strictly positive function $g_t : \mathbb{R}^3 \to \mathbb{R}_+$ then the product $g_t \cdot F_t$ is also a level set function for Ω_t. One canonical choice could be to take $F_t(x)$ as the *signed distance function* of Ω_t. This function is mostly smooth and satisfies $\|\nabla F_t(x)\| = 1$ for almost all $x \in \mathbb{R}^3$, but has the drawback that it is not differentiable on a non-empty (though measure zero) set called the *medial axis* of $\partial\Omega_t$ [OF02]. Our formulation, however, is not tied to a particular choice of level set function nor is it sensitive to the presence of this type of non-differentiability. The only assumptions we make are as follows. The surface $\partial\Omega_t$ is piecewise smooth and has a level set function F_t defined in a neighborhood of $\partial\Omega_t$ of uniform size. Moreover, F_t has uniformly bounded difference quotients in space and in time. Finally, $\|\nabla F_t\| > 0$ except possibly at isolated points.

3.2. Changing Topology

One of the chief advantages of the level set representation for time-varying geometry is that changes of topology can be handled seamlessly and without the need for complex and expensive geometry

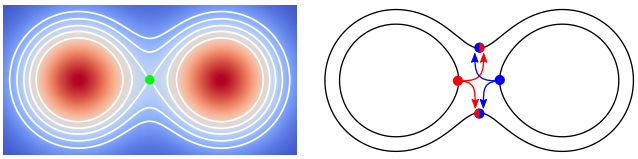


Figure 2: (left) The level set function has a critical point near a moment of topology change. (right) Tracking ambiguity.

processing operations such as temporally coherent remeshing. Applications which have leveraged this feature with great success include modeling interface pheonomena in fluid flow (many examples can be found in [OF02]) and topology optimization of continuum structures in engineering (e.g. [AJT04] and [WWG03]).

The difficulty posed by changing topology, from our perspective, is that when the topology of the zero-level set of a family functions F_t changes, it must be the case that ∇F_t vanishes somewhere on the level set (if this were not so, we would be in contradiction to the implicit function theorem [Mun91]), see Figure 2.

We nevertheless would like our level set tracking algorithm to be applicable during a change in topology. We note that tracking points on a family of level sets undergoing a change in topology is ill-posed and highly dependent on external considerations. For instance, Figure 2 shows an ambiguity that bedevils any attempt to track points on a pair of merging spheres based on surface geometry alone; however, a resolution is possible e.g. if the spheres represent water droplets and our model includes hydrodynamic effects. Therefore we will develop our algorithm with sufficient flexibility to be able to handle changes of topology by allowing the user to incorporate external tracking strategies in the space-time neighbourhood of the locus of topology change. We address this in §5.6.

3.3. Tracking a Family of Level Sets

We would like to track the motion of an arbitrary point on the family of surfaces $\partial\Omega_t$. What this means is that we are able to construct a smooth family of mappings (called a *flow*) of the form $\phi_t : \partial\Omega_0 \to \mathbb{R}^3$ such that $\phi_t(x_0) \in \partial\Omega_t$ for each $x_0 \in \partial\Omega_0$ and $\phi_0 = \textit{identity}$. Therefore, for any point $x_0 \in \partial\Omega_0$ we get a smooth trajectory defined by $x(t) := \phi_t(x_0)$ that tracks x_0 over time.

The choice of ϕ_t is highly non-unique since it is equivalent to finding *correspondences* between the surfaces at different times. If Ω_t moves rigidly, then there may be well-defined "natural" correspondences. But if the motion of Ω_t involves deformation, then correspondences cease to be well-defined. Nonetheless, in many situations we seek the "most natural" correspondences between the surfaces. The challenge becomes how to define this notion rigorously. In this paper, we propose a variational method that singles out a unique ϕ_t for any motion of Ω_t and is optimally applicable when Ω_t moves near-rigidly.

The choice of ϕ_t can not be made arbitrarily since geometric constraints must be satisfied. The level set representation provides a convenient way to express these. We rephrase the condition $\phi_t(x_0) \in \partial\Omega_t \ \forall (x_0,t)$ as $F_t(\phi_t(x_0)) = 0$ and differentiate in t. This yields

$$\frac{\partial F_t}{\partial t} \circ \phi_t(x_0) + \nabla F_t \circ \phi_t(x_0) \cdot \frac{\partial \phi_t(x_0)}{\partial t} = 0\,. \tag{1}$$

Here, the derivative $\frac{\partial \phi_t(x_0)}{\partial t}$ is the velocity vector field of the trajectory $x(t) := \phi_t(x_0)$. Since $\nabla F_t(x)$ is orthogonal to the level set containing x, thus (1) constrains the normal component of the velocity and leaves the tangential component unconstrained. Any choice of ϕ_t must satisfy this constraint. Conversely, it is the freedom in the tangential component, illustrated in Figure 1b, that leads to our inability to determine ϕ_t. In this light, our method for determining ϕ_t amounts to making a choice of the tangential component of the velocity while constraining the normal component via (1).

3.4. Eulerian Velocity

Since we wish to construct ϕ_t from its velocity, we proceed as follows. Let $\mathcal{U} \subseteq \mathbb{R}^3$ be a large bounded domain containing all Ω_t. Consider an *extended* mapping $\Phi_t : \mathcal{U} \to \mathbb{R}^3$ such that the restriction of Φ_t to $\partial\Omega_0$ is the desired ϕ_t. We introduce the *Eulerian velocity* of this mapping, defined by $V_t(x) := \frac{\partial \Phi_t}{\partial t} \circ \Phi_t^{-1}(x)$, so that $V_t(x)$ is the velocity at x of the trajectory passing through x at time t. Therefore the trajectories $x(t) := \Phi_t(x_0)$ can be seen as solutions of the system of ordinary differential equations (ODEs)

$$\frac{dx}{dt} = V_t(x) \quad \text{and} \quad x(0) = x_0\,. \tag{2}$$

The construction of ϕ_t can now be carried out by first specifying V_t and then integrating (2) for Φ_t and restricting the result to $\partial\Omega_t$.

To have well-posed constraints in the above formulation, we must constrain V_t everywhere in $\mathcal{U}$. We make the assumption that the extended mapping Φ_t preserves *all* level sets of the family of functions F_t, and not just the zero level set; i.e. $F_t(\Phi_t(x)) = \textit{constant}$ for all $(x,t) \in \mathcal{U} \times [0,1]$. This is reasonable because in many practical cases, e.g. signed distance functions or level set functions defined by Boolean operations, the nearby level sets to the geometry of interest move in conjunction with each other. Differentiating, we find

$$\frac{\partial F_t}{\partial t} + \nabla F_t \cdot V_t = 0\,. \tag{3}$$

This equation constrains the normal component V_t to be

$$V_t^{\perp} = -\frac{\partial F_t}{\partial t} \frac{\nabla F_t}{\|\nabla F_t\|}\,. \tag{4}$$

It remains to determine V_t in its entirety. We will explain our proposed method for this after a brief introduction to *approximate Killing vector fields*.

3.5. Approximate Killing Vector Fields

In Riemannian geometry, one characterizes the Eulerian velocity of a rigid motion via the *Killing equation*, named after the mathematician W. Killing (1847–1923) [dC92]. This is done as follows. We define the linear partial differential operator P : *Vector fields* $\to$ *Symmetric matrix fields* known as the *Killing operator* by

$$P(V) := DV + \left[DV\right]^{\top}.$$

A vector field V satisfies the Killing equation when $P(V) = 0$, and then it is called a *Killing vector field*. The connection to rigid motions is given by the following result.

Theorem 1. *Let $R_t : \mathbb{R}^3 \to \mathbb{R}^3$ be a family of rigid motions of $\mathbb{R}^3$ of the form $R_t(x) := \mathcal{O}_t x + p_t$ where $\mathcal{O}_t \in \mathrm{SO}(3)$ is an orthogonal matrix with unit determinant (the rotational part of R_t) and $p_t \in \mathbb{R}^3$ is a vector (the translational part of R_t). Then its Eulerian velocity at any time t is a Killing vector field. Conversely, suppose $V : \mathbb{R}^3 \to \mathbb{R}^3$ is a Killing vector field. Then, there is an antisymmetric matrix $A \in \mathbb{R}^{3\times 3}$ and a vector $b \in \mathbb{R}^3$ so that $V(x) = Ax + b$. Moreover, any sufficiently smooth family of solutions of this equation is the Eulerian velocity of a family of rigid motions.*

We now define an *approximate Killing vector field* (AKVF) as a *least-squares* solution of the Killing equation subject to zero or more application-dependent constraints. In the absence of constraints, the resulting vector fields satisfy the Killing equation exactly and thus generate rigid motions. In the presence of constraints, the resulting vector fields are as close as possible to being generators of rigid motions. For example, in [BCBSG10, SBCBG11b] the constraints were geometric and arose indirectly because the authors formulated their AKVF problem on a curved surface; whereas in [SBCBG11a], the constraints were prescribed externally by the user.

3.6. AKVFs for Level Set Tracking

Finally we are able to state our method for determining the velocity V_t. We require V_t to be an AKVF as defined above with the constraint (3) on the normal component of V_t. In this way, we produce vector fields that are as close to generating rigid motions as possible, subject to the level set tracking condition.

Definition 2. Let $\mathcal{U}$ be a bounded domain in $\mathbb{R}^3$ with smooth boundary containing a moving zero-level set generated by the family of functions $F_t : \mathcal{U} \to \mathbb{R}$. Then we define the velocity vector field for tracking these level sets as the AKVF

$$\left.\begin{aligned} V_t &:= \arg\min_{V\in H^1} \int_{\mathcal{U}} \|P(V)\|^2 \\ \text{s.t. } & \frac{\partial F_t(x)}{\partial t} + \nabla F_t(x)\cdot V(x) = 0 \;\forall x \in \mathcal{U} \end{aligned}\right\} \forall t \in [0,1] \qquad (5)$$

where we minimize over $H^1 := H^1(\mathcal{U}, \mathbb{R}^3)$, the set of L^2-integrable vector fields on $\mathcal{U}$ with L^2-integrable weak first derivatives.

Note. An important caveat is that we require $\nabla F_t(x) \neq 0$ for all $x \in \mathcal{U}$ to ensure that the constraints are non-degenerate. This is of course a very restrictive requirement to guarantee in practice (e.g. any differentiable level set function for Ω_t must have at least one critical point corresponding to the global minimum). However, we will see below (§3.10) that it is possible to slightly reformulate the optimization problem (5) in such a way that this requirement is obviated. In the mean time, we proceed as if this were the case.

Note. In Definition 2 we use the Frobenius norm $\|P(V)\|^2 := \mathrm{Tr}\big(P(V)[P(V)]^\top\big)$. This could be replaced with a more general quadratic of the form $\mathrm{Tr}\big(P(V)M_1[P(V)]^\top M_2\big)$, where $M_1, M_2 \in \mathbb{R}^{3\times 3}$ could incorporate anisotropy or re-weighting, for instance. We do not exploit this freedom here, though it does suggest an interesting direction for future work.

3.7. Constraint Handling

We handle the constraints using a *reduced formulation*. That is, we incorporate the constraints directly into the space of admissible vector fields by writing $V_t = V_t^\perp + V_t^\parallel$ where $V_t^\perp$ is normal to the level sets of F_t and is given by (4), while $V_t^\parallel$ is parallel to the level sets of F_t and is determined by solving

$$V_t^\parallel := \arg\min_{V\in H^1_\parallel} \int_{\mathcal{U}} \|P(V_t^\perp + V)\|^2 \quad \forall t \in [0,1] \qquad (6)$$

where $H^1_\parallel$ is the set of admissible vector fields satisfying $\nabla F_t(x)\cdot V(x) = 0$ for all $x \in \mathcal{U}$. This problem is unconstrained since we can derive a basis of tangent vectors satisfying this equation at each x.

Note. The requirement $\nabla F_t(x) \neq 0$ is now explicit since $\|\nabla F_t\|$ appears in the denominator of the expression defining $V_t^\perp$.

3.8. Properties of AKVFs

Convexity. The operator P is a linear, first-order partial differential operator and the norm $\|\cdot\|^2$ is positive definite and quadratic. The constraints are linear. Thus, the optimization objective in (6) is a convex quadratic function bounded below by zero. By standard results in the calculus of variations, a minimizer exists in the space of vector fields $H^1(\mathcal{U}, \mathbb{R}^3)$ [JLJ98].

Ellipticity. We show in the supplementary material that minimizers of (5) and (6) satisfy a partial differential equation (PDE):

$$\begin{aligned} P^*P(V) + \lambda\nabla F_t &= 0 \qquad \text{in } \mathcal{U} \\ \nabla F_t \cdot V &= -\frac{\partial F_t}{\partial t} \qquad \text{in } \mathcal{U} \\ N_{\partial\mathcal{U}} \cdot P(V) &= 0 \qquad \text{on } \partial\mathcal{U}. \end{aligned} \qquad (7)$$

Here, P^* is the matrix divergence, given component-wise by $[P^*(M)]^i := \sum_j \frac{\partial M_{ij}}{\partial x^j}$ where M_{ij} are the components of M and $N_{\partial\mathcal{U}}$ is the outward unit normal vector field of $\partial\mathcal{U}$. This is a system of linear partial differential and algebraic equations for V and a Lagrange multiplier function $\lambda : \mathcal{U} \to \mathbb{R}$.

The second-order, linear partial differential operator $P^*P(V) := \mathrm{div}\big(DV + [DV]^\top\big)$ is a particularly well-behaved *elliptic operator* [CW98]. We can think of P^*P as a vectorial analog of the well-known Laplace operator. In fact, if V^j are the components of V then the components $P^*P(V)$ are

$$[P^*P(V)]^i = -\Delta V^i - \frac{\partial}{\partial x^i}\mathrm{div}(V),$$

where Δ is the Laplace operator. Thus the solutions of (7) enjoy some properties of solutions of the Laplace equation (though notably not all, e.g. the maximum principle does not hold). In particular, we can expect solutions of (7) to enjoy a *smoothing property*. For instance, we expect reasonable output when F_t is only continuous with uniformly bounded difference quotients.

Null space. We now identify the null space of (7). Suppose V, λ are such that the right hand side of (7) vanishes. Integrating the inner product of the first equation with V by parts over $\mathcal{U}$ shows $\int_{\mathcal{U}} \|P(V)\|^2 = 0$ using the boundary condition. Thus $P(V) \equiv 0$ everywhere in $\mathcal{U}$; in other words, V is a Killing vector field. This in

turn shows that $\lambda = 0$ whenever ∇F_t is non-zero (which will turn out to be everywhere in $\mathcal{U}$ according to the small reformulation we will make in §3.10). Finally, the constraint $\nabla F_t \cdot V \equiv 0$ shows that this Killing vector field is everywhere tangent to the level sets of F_t.

Therefore the null space of (7) is non-trivial only for special domains and level set functions — those for which there exists a Killing vector field that is tangent to all level sets. The only examples satisfying this condition are shapes possessing *rotational symmetries* such as a ball or the interior of a compact surface of revolution (compact shapes with translational symmetries do not exist).

Theorem 3. *The minimization problem (6) possesses a unique solution unless Ω_t possesses a rotational symmetry and we represent Ω_t by a level set function that is invariant under this symmetry.*

We note that the limitation expressed in this theorem is a rare event in practice. The failure to compute a reasonable vector field in this case is ultimately due to the fundamental limitation that implicit surfaces do not carry information about points on the surface (i.e. the inability to distinguish a stationary sphere from a rotating sphere). We discuss how our algorithm addresses this issue in §5.3.

2D versus 3D. AKVFs can be defined equally well for a family of moving domains in $\mathbb{R}^2$ specified as the zero sub-level sets of functions $F_t : \mathbb{R}^2 \to \mathbb{R}$. Some examples and applications that we will present in the sequel are in 2D; the relevant mathematics and computational framework are essentially the same as in 3D.

3.9. Behavior With Respect to Rigid Motions

An important feature of the problem (5) is that it exactly captures rigid motions. See the supplementary material for proof.

Theorem 4. *Suppose an object moves rigidly, i.e. $\Omega_t := R_t(\Omega)$ for some time-varying family of rigid motions and reference geometry Ω, and we describe this motion using the level set function $F_t := F \circ R_t^{-1}$ where F is a level set function for Ω. Then the Eulerian velocity $V_t := \frac{dR_t}{dt} \circ R_t^{-1}$ solves the constrained minimization problem (5).*

3.10. Narrow Band Reformulation

It is necessary to pose the optimization problem for the vector field V_t in a domain $\mathcal{U}$ that strictly contains $\partial\Omega_t$ in order to achieve a formulation governed by the well-posed elliptic partial differential equation 7. However, there is nothing in our formulation so far that specifies the *size* of $\mathcal{U}$, which we are therefore free to choose as convenient. Since we are really only interested in tracking the zero-level set $\partial\Omega_t$, it makes sense from a numerical perspective to perform calculations only on a *narrow band* around $\partial\Omega_t$. Moreover, since points where $\nabla F_t(x) = 0$ cause the constraints to degenerate, these should be excluded as well. Recall that we have made the reasonable assumption that ∇F_t has only isolated zeros.

Therefore a small reformulation of (6) yields greater efficiency and applicability. Rather than posing the optimizations for each t in the same large domain $\mathcal{U}$, we need only pose the problem at time t in the narrow-band domain $\mathcal{U}_\varepsilon(t) := U_1 \cap U_2$ where

$$U_1 := \{x \in \mathbb{R}^3 : \operatorname{dist}(x, \partial\Omega_t) < \varepsilon\}$$

$$U_2 := \{x \in \mathbb{R}^3 : \|x - \bar{x}\| > \bar{\varepsilon} \ \ \forall \bar{x} \text{ s.t. } \nabla F_t(\bar{x}) = 0\}$$

and $\operatorname{dist}(x, \partial\Omega_t)$ is the distance of $x \in \mathbb{R}^3$ to $\partial\Omega_t$, while $\varepsilon, \bar{\varepsilon}$ are small parameters specified *a priori*. We also only integrate the ODE (2) for trajectories that begin on $\partial\Omega_0$ and remain in the narrow bands over time. We should point out that the narrow-band approach is common in applications concerning level sets, e.g. [AS99], and can make use of specialized data structures, e.g. [Mus13].

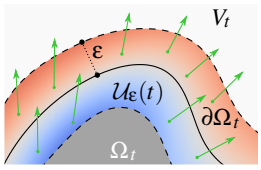


Figure 3: Narrow band

Note. In practice, we find that choosing ε as a small fraction of the diameter of $\partial\Omega_t$ and $\bar{\varepsilon}$ as a small fraction of ε are effective choices for a wide range of shapes. In §5.3 we demonstrate the insensitivity of our results to the width of the narrow band.

4. Discretization

4.1. Discrete AKVFs

Problem (6) admits a straightforward discretization via finite elements. In the end, this yields a linear system of equations for the expansion coefficients of V. The details are as follows; we fix a particular time t and suppress the t-subscript for brevity.

Discretization of the domain. Suppose that the geometry of interest is contained within a background regular tetrahedral mesh $\mathcal{T}$ with vertices $x_1, \ldots, x_N$, obtained by subdividing each cell of a cubical grid into 6 tetrahedra (i.e. the Freudenthal subdivision [Fre42] in Figure 4). We assume that values of the level set function are given at the vertices and the space and time derivatives are also given there either by centered finite differences or via analytical formulas. We define a discrete narrow band of widths ε and $\bar{\varepsilon}$ for $\partial\Omega$ as the union of tetrahedra

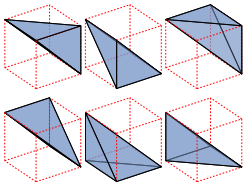


Figure 4: Tetrahedra.

$$\mathcal{U}_\varepsilon := \bigcup \{[y1, y2, y3, y4] \in \mathcal{T} : \operatorname{dist}(y_i, \partial\Omega) \le \varepsilon \ \forall i\} \setminus \mathcal{C}_{\bar{\varepsilon}}$$

where $\mathcal{C}_{\bar{\varepsilon}}$ is the set of tetrahedra whose vertices are within $\bar{\varepsilon}$ of the discrete critical points of F. We compute the distance function to $\partial\Omega$ using the fast marching method as described in [Set99].

Discretization of vector fields. We work with piecewise-linear vector fields on $\mathcal{U}_\varepsilon$. A basis consists of the vector fields $\xi_i e_s$ for $i = 1, \ldots, N$ and $s = 1, 2, 3$ where e_s is the s^{th} standard basis vector and ξ_i is the piece-wise linear function satisfying $\xi_i(x_j) = \delta_{ij}$.

Incorporating the constraints. Write $V := \sum_i \sum_{s=1}^3 v_{is} \xi_i e_s$. We impose the constraint (5) at each vertex; therefore v_{is} must satisfy

$$\sum_{s=1}^{3} v_{is} \frac{\partial F}{\partial x^s}(x_i) + \frac{\partial F}{\partial t}(x_i) = 0 \qquad \forall\, x_i\,. \tag{8}$$

By construction of the narrow band, at least one of the spatial derivatives is nonzero so these 1×3 systems are solvable. Thus

$$v_{is} := \sum_{s'=1}^{2} a_{is'} z_{iss'} + w_{is} \tag{9}$$

where $a_{i1}, a_{i2} \in \mathbb{R}$ are real coefficients that will become the new unknowns and $z_{iss'} \in \mathbb{R}$ are chosen to satisfy $\sum_{s=1}^{3} z_{iss'} \frac{\partial F}{\partial x^s}(x_i) = 0$ for $s' = 1,2$ (thus $\sum_{s=1}^{3} z_{iss'} e_s$ are chosen to be tangent to the level set at x_i) and finally, $w_{is} \in \mathbb{R}$ is the normal component of the velocity

$$w_{is} := -\frac{\partial F}{\partial x^s}(x_i) \frac{\partial F}{\partial t}(x_i) / \|\nabla F(x_i)\|^2 . \tag{10}$$

Discretization of the reduced system. In the supplementary material, we derive the discrete optimality conditions relating the unknown tangential components a_{is} and the normal component (10). In summary, this derivation shows that we require the following objects computable from the input data and background mesh:

- The components of a stiffness matrix

$$K_{ijst} := \sum_{T \in R(i,j)} \frac{2A(i,T)A(j,T)}{9\mathrm{Vol}(T)} \left(n_s(i,T) n_t(j,T) + \delta_{st} \right) \tag{11}$$

  where $R(i,j)$ is the one-ring of tetrahedra containing edge $[x_i, x_j]$ and $n(i,T)$ is the inward-pointing unit vector normal to the face opposite vertex i of tetrahedron T with area $A(i,T)$.
- A vector $w \in \mathbb{R}^{3N}$ written as $w := (w_1^\top, w_2^\top, w_3^\top)^\top$ and $w_s^\top = (w_{1s}, \ldots, w_{Ns})$ is computed per-vertex using (10).

Then we define block matrices $Z \in \mathbb{R}^{3N \times 2N}$ and $K \in \mathbb{R}^{3N \times 3N}$ as

$$Z := \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \\ Z_{31} & Z_{32} \end{pmatrix} \quad \text{and} \quad K := \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{12}^\top & K_{22} & K_{23} \\ K_{13}^\top & K_{23}^\top & K_{33} \end{pmatrix}$$

where $Z_{ss'} := \mathrm{diag}([z_{iss'}]) \in \mathbb{R}^{N \times N}$ and the components of the matrices $K_{st} \in \mathbb{R}^{N \times N}$ are just K_{ijst}. The matrix Z is block-diagonal, while (11) shows that K is sparse with $K_{ijst} \neq 0$ when (i,j) is an edge in $\mathcal{T}$. Finally, we solve

$$Z^\top K Z a + Z^\top K w = 0, \tag{12}$$

where $a := (a_1^\top, a_2^\top)^\top$ with $a_{s'}^\top := (a_{1s'}, \ldots, a_{Ns'})$ is the unknown in $\mathbb{R}^{2N}$. We use the conjugate gradient algorithm to solve (12).

Discrete Killing vector fields. We have seen that Killing vector fields are affine vector fields of the form $V(x) := Ax + b$ where A is an antisymmetric matrix and b is a vector. An affine vector field is equal to its piecewise linear interpolation. Since we have used piecewise linear shape functions for the discretization of the stiffness matrix above, we can therefore assert that the discretized Killing vector fields belong to the null space of the discrete stiffness matrix. Consequently, an analogue of Theorem 4 holds and the discrete constrained optimization problem can reproduce rigid motions up to error in the time discretization of the level set function.

4.2. Discrete Vector Field Integration

The solution of the discrete system described in the previous section, applied at each time t, yields a family of piecewise linear vector fields V_t, each defined on a narrow band $\mathcal{U}_\varepsilon(t)$. We now describe a method for integrating the ODE (2) for trajectories starting near $\partial\Omega_0$ in the narrow band $\mathcal{U}_\varepsilon(0)$. We will use a forward time integration method. We would like to ensure, however, that if V_t generates a family of rigid motions then the integrated trajectories are exactly correct. In [SBCBG11a], this was accomplished by choosing the trajectories to be *logarithmic spirals*. But there is no analogue of these trajectories in three dimensions, so we proceed differently: we use a *first-order Euler exponential integrator* [HLW06].

To be precise, we discretize the time interval $[0,1]$ into M time steps of size $\Delta t := 1/M$ and let $t_m := m\Delta t$. We approximate the ODE (2) for the trajectory starting at $x_0 \in \mathcal{U}_\varepsilon(0)$ by a piecewise continuous sequence of arcs of the form $t \mapsto x_m(t)$ for $t \in [0, \Delta t]$ and $m = 1, \ldots, M-1$. The equation for the $(m+1)^{st}$ arc derives from the first-order Taylor expansion of (2) at $(x_m(\Delta t), t_m)$, namely

$$\begin{aligned} \frac{dx_{m+1}}{dt} &= V_m + \dot{V}_m (t - t_m) + A_m (x_{m+1} - p_m) \\ x_{m+1}(0) &= p_m \end{aligned} \tag{13}$$

where

$$p_m := x_m(\Delta t) \qquad V_m := V_{t_m}(p_m)$$
$$A_m := [DV_{t_m}(p_m)]^{antisym} \qquad \dot{V}_m := \left. \frac{\partial V_t(p_m)}{\partial t} \right|_{t=t_m} .$$

In our implementation, we compute DV_m and $\dot{V}_m$ using centered finite differences. Note that we have replaced the full derivative DV_m appearing in the first-order Taylor expansion with its antisymmetric part A_m. Since V_m is an AKVF for which the symmetric part $P(V_m)$ is small in the least-squares sense by definition, and since we expect the motion we are tracking not to depart too significantly from rigid, then we expect A_m to be a good approximation of DV_m.

We can find an explicit solution for (13) based on the Rodrigues Formula for the exponential of an antisymmetric matrix. Let $\omega_m = \frac{1}{2}\|A_m\|_{Fro}$. Then after some work,

$$x_{m+1}(t) = \begin{cases} p_m + tV_m + \frac{t^2}{2}\dot{V}_m & \text{if } \omega_m = 0 \\ p_m + E_{1m}(t) \cdot V_m + E_{2m}(t) \cdot \dot{V}_m & \text{if } \omega_m \neq 0 \end{cases} \tag{14}$$

where the 3×3 matrices $E_{1m}(t)$ and $E_{2m}(t)$ are:

$$E_{1m}(t) := tId + \frac{1 - \cos(\omega_m t)}{\omega_m^2} A_m + \frac{t\omega_m - \sin(\omega_m t)}{\omega_m^3} A_m^2$$

$$E_{2m}(t) := \frac{t^2}{2} Id + \frac{t\omega_m - \sin(\omega_m t)}{\omega_m^3} A_m + \frac{\frac{1}{2}t^2\omega_m^2 - 1 + \cos(\omega_m t)}{\omega_m^4} A_m^2 .$$

Note that if $\omega_m \to 0$ then the second option of (14) reduces to the first option, which is a standard Euler step. We conclude this section with an important result, showing that our integrator reproduces uniform rigid motions exactly.

Theorem 5. *Let $R_t : \mathbb{R}^3 \to \mathbb{R}^3$ be a one-parameter family of uniform rigid motions. Then the trajectories generated by equations (14) and the trajectories $x \mapsto R_t(x)$ are identical.*

Proof. Let $V_t(x) := \frac{\partial R_t}{\partial t} \circ R_t^{-1}(x)$ be the Eulerian velocity of R_t. Since R_t is a uniform rigid motion, $V_t(x)$ is affine in both x and t. Thus it equals its first-order Taylor expansion at each (x,t). □

4.3. Stabilization

To prevent particles from drifting from the zero level set after each integration step, we project particles back to $\partial\Omega_t$ by applying

$$x \leftarrow x - \frac{F_t(x) \nabla F_t(x)}{\|\nabla F_t(x)\|^2} .$$

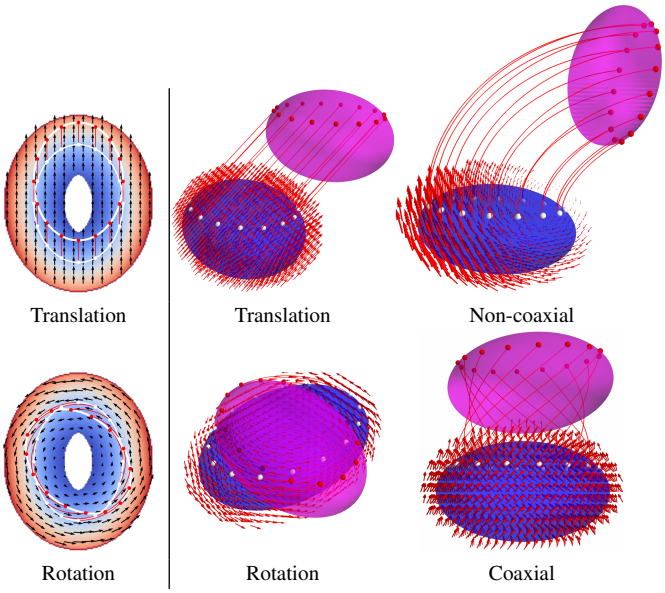


Figure 5: AKVFs and tracked particles on a rigidly moving ellipse in 2D and 3D. Our method recovers these rigid motions exactly.

This formula derives from the first-order Taylor expansion of nearest-point projection. One iteration suffices for our purposes since the deviation from the zero level set is always small. Note that since we have excluded points where $\|\nabla F_t\|$ vanishes from the narrow band, the projection above is always well-defined. But we should thus expect instabilities during topology change; see §5.6.

5. Results and Applications

All our test cases are either subsets of the unit square $[0,1]^2$ in 2D sampled at various resolutions (usually 200×200) or the unit cube $[0,1]^3$ sampled at various resolutions (usually $75 \times 75 \times 75$). These are narrow-band level sets of width $\varepsilon = 3/64$ unless otherwise indicated. We compute derivatives either using analytically defined level set functions or via finite differences. Since the former are more common, we only point out the latter below; in any case, we demonstrate in §5.4 that our results are robust to discretization. Our implementation is in Python with `numpy`; visualizations are generated with `matplotlib` in 2D and `mayavi` in 3D. We compute signed distance functions using the fast marching method from `scikit-fmm`.

5.1. Linear Rigid and Non-Rigid Motions

Rigid motions. A key benefit of our algorithm is that it captures rigid motion exactly. Figure 5 shows the first and last frames in 50-frame sequence of an ellipse undergoing assorted rigid motions. In 2D we superpose the initial and final zero-level sets in the first frame, along with the computed AKVF and the trajectories of a few particles placed on the initial zero-level set. In 3D we show the zero-level sets in different colors (blue=first and purple=last), as well as the computed AKVF in the first frame and particle trajectories. In all cases, our algorithm reproduces the rigid motion exactly to within solver tolerance. The rotation with non-coaxial translation

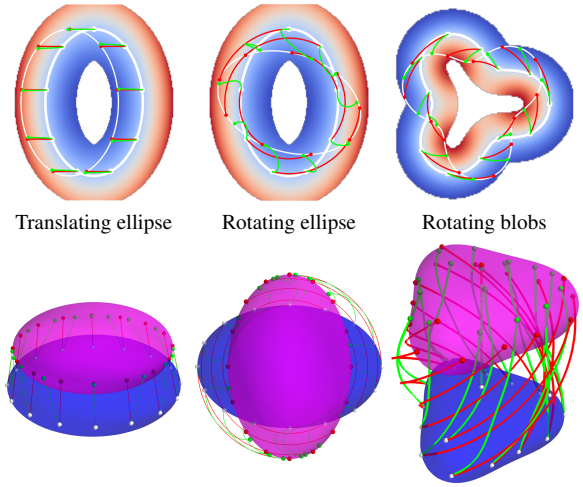


Figure 6: Comparison with [SS11] for rigid motions in 2D and 3D. Red curves are ground truth motions, reproduced by our algorithm. Green curves are motions produced by [SS11].

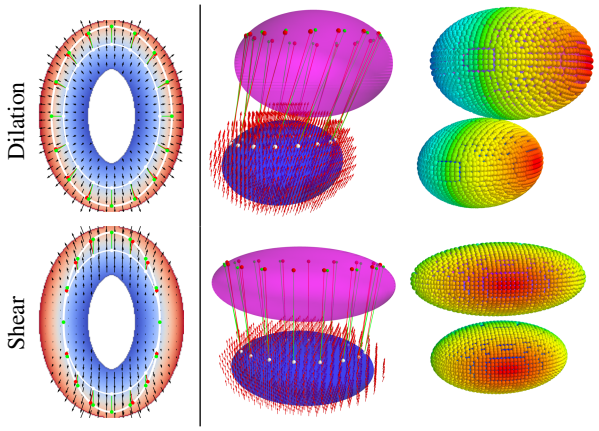


Figure 7: AKVFs and tracked particles for linear non-rigid motions of an ellipse in 2D and 3D, showing tracked paths (red) and linear motion paths (green). In the 3D case we show a small set of particle trajectories (left) and the initial and final positions of a sampling of particles colored by their x-coordinate in the initial frame (right).

example is notable because its vector field varies in time and space, thus making full use of our integrator's capabilities. Although these results use analytical derivatives, their accuracy is retained when we use finite-difference derivatives, shown in §5.4.

Figure 6 compares with analogous results using [SS11]. Since this method's formulation does not account for rotation, its accuracy suffers predictably on motions with a rotational component. In particular, combining rotation and coaxial translation causes considerable difficulty. We did not compare with [FMM13], because unlike [SS11] it is highly unsuitable for level set data with discrete derivatives due to higher-order differentiation. But even if we used this method with analytical derivatives, its curvature-based tracking model is ill-suited to tracking surfaces that deform over time, as small non-rigid deformations can have large changes in curvature.

Linear non-rigid motions. We perform two experiments with linear motion of the form $F_t(x) := \exp(At) \cdot x$ where A is a constant but not antisymmetric matrix (so $\exp(At)$ is no longer a rotation). Examples include shears and dilations. We expect the computed AKVFs and tracking results to exhibit simple departures from the linear motions in question, as these motions are non-rigid. In each case we show the first and last frames in a 50-frame sequence.

Figure 7 shows AKVFs and tracking for an ellipse undergoing shearing or dilation using the same visualization as for Figure 5 (in 3D we added a translational component to cause the ellipses to pull apart for greater clarity). Tracked paths are shown in red while paths corresponding to the linear motions are shown in green; well-bounded discrepancies between these paths are clearly visible. In the 3D case we track a collection of colored particles generated by farthest point sampling to demonstrate that the particle distribution behaves reasonably over time. We observe that [SS11] has equivalent behavior in these cases.

Figure 8 examines the effect of shearing and dilation for a 50-frame sequence of level sets represented by the signed distance function sampled on an $80 \times 80 \times 80$ background grid. We compute derivatives via finite differences. We show the AKVF in the first frame and an alternate visualization of tracking results. That is, we track vertices of a textured mesh at $t = 0$ to render the moving model in several frames. The last column shows error as deviation from the linear motion undergone by the vertex positions in the last frame. Our technique tracks the rotating model accurately and produces reasonable tracking results for dilation and shearing.

The "squash" test, wherein the cow is compressed horizontally against the side of the grid, illustrates drawbacks of our method. First, once the model is compressed enough, the grid resolution is insufficient to track sharp features like the horns or to distinguish between the legs. Second, when the level set approaches the grid boundary, the neighborhood $\mathcal{U}_\varepsilon(t)$ must be relatively thin. Even so, our algorithm tracks large-scale deformation of the cow, with some artifacts where motion could not be resolved. We do not include comparisons in this experiment to [SS11], as we were unable to produce reasonable results using a relatively complex signed distance function sampled on a grid rather than known analytically.

5.2. Non-rigid deformations

We show the results of two experiments in which we track non-linear, non-rigid, procedurally-generated deformations. Figure 10 shows AKVFs and tracked paths for a shape moving and deforming through 50 frames. In this example, an implicit surface moves vertically while rotating and bulging out and in; the level set function is a sum of three Gaussians with moving centers and constant variances. Points are successfully tracked through this challenging sequence. Since the shape returns to its original configuration modulo 120° rotation and vertical shift, we have a notion of "ground truth"; from this perspective, the tracked points exhibit 0.5% error on average from their expected positions. The inset Figure 9 shows tracked trajectories pulled back to the initial frame in green and the original particle locations in white.

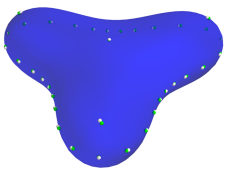


Figure 9: Pullback

Figure 11 shows AKVFs and tracked paths for an articulated bar as it bends about non-coplanar axes at two joints. The bar is represented as a sequence of signed distance functions and we use finite-difference derivatives. Points remain at the expected locations even as it bends and deforms in the joint regions. Due to the degree of deformation in the joints, it is not clear how to pull back many of the particles to the initial frame; thus we do not show an analogue of Figure 9. Visually, however, the results are reasonable.

Finally, in Figure 12 we track a large collection of particles on each of the two shapes above and visualize them using the technique of Figure 7 (far right) in order to show that the particle distribution behaves reasonably throughout the deformation.

5.3. Geometric Robustness

Narrow band size. Figure 13 shows an experiment illustrating our method's insensitivity to the choice of narrow band size ε. For clarity of visualization we show the experiment in 2D but observe similar robustness in 3D. We show vector fields and tracked points at three frames of a 50-frame sequence of motions of a two-dimensional implicit shape undergoing pinching and rotation with varying choices of ε. The computed vector field V_t is nearly identical near the level set $\partial\Omega_t$ in each example, and the positions of the tracked points agree up to several digits of precision. We only observe significant degradation below $\varepsilon = 1/128$ or a diameter less than 2 grid cells.

The choice of level set function. Figure 14 shows an experiment where we test the dependence of our method on the choice of level set function used to describe the evolving geometry. For clarity of visualization we show the experiment in 2D but observe similar robustness in 3D. We show level sets, vector fields and tracked points for the motion in the first and last frame of a 25-frame sequence of an implicit shape undergoing pinching only. The level set functions are: the function F_t equal to a sum of Gaussians; the function $F_{1,t}(x) := p(F_t(x))$ where $p(z) := z^3 + \frac{3}{2}z$; and the function $F_{2,t}(x) := g(x)F_t(x)$ where $g(x) := \frac{1}{10} + \|x\|^2$.

The function $F_{1,t}$ simply rescales the values of F_t and thus all level sets continue to exhibit similar deformations. Consequently, we observe that the tracking accuracy of $F_{1,t}$ relative to F_t is extremely high. However, the function $F_{2,t}$ has the same zero-level set as F_t but its nonzero-level sets exhibit different deformations in each frame. The tracking accuracy is correspondingly reduced, though in absolute terms the results are stable and reasonable.

To address the question of which level set function is best, we note that a canonical choice is always available, i.e. the signed distance function. Furthermore, we recall that the problem of tracking a non-rigidly deforming family of implicit surfaces is ill-posed and thus one expects the need for additional input in order to resolve all ambiguities. It is an interesting direction of future research to optimize the choice of level set function towards specific applications.

Discontinuous derivatives. Figure 15 shows an experiment where we test the accuracy of our method when the derivatives of the level set function possess discontinuities. This is an important case because the signed distance function of a shape is non-differentiable on the medial axis. But we expect this not to be a critical issue because

Figure 8: We apply three large, extrinsic motions to a textured mesh and track its vertices over time. We show an example of the vector field V on slice $\mathcal{U}_\varepsilon(t)$, example frames of the tracked mesh, and displacement error between the tracked mesh and the linear motion in the final frame. (Cow model courtesy K. Crane.)

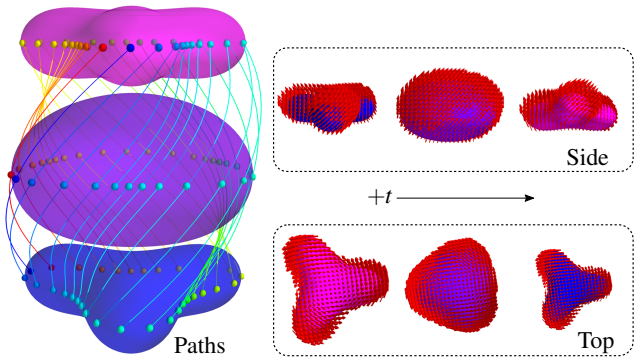


Figure 10: (left) Paths of points on an implicit surface moving vertically, rotating, and deforming; colors represent the surfaces at frames 0, 25, and 50. (right) AKVFs for the motion, from two views.

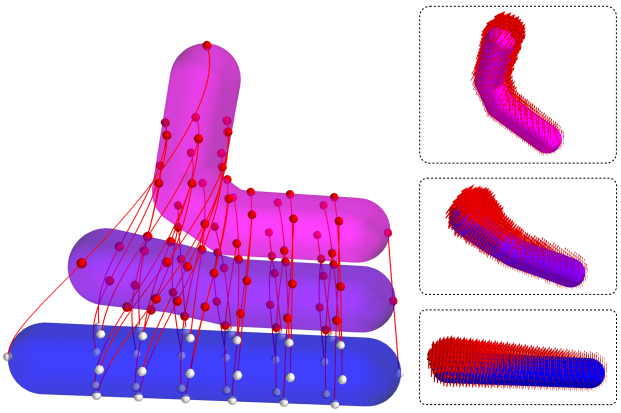


Figure 11: (left) AKVFs for frames 0, 25 and 50 in an animated sequence of an articulated bar; (right) tracked paths of several points on the bar through the animation. Our algorithm generates expected trajectories even though the bar deforms non-rigidly.

our algorithm computes vector fields by solving an elliptic PDE and thus enjoys the benefits of its smoothing properties. Indeed, our approach is theoretically guaranteed to be stable if the input possesses the smoothness of a signed distance function. This is confirmed by our experiments: the left panel of Figure 15 is coloured with the time derivative of the signed distance function of a rotating square and we also show its gradient vector field (the discontinuities are clearly visible). In the right panel, we show the final frame of the motion of the square rotating through 45° in a 50-frame sequence. The AKVF and tracking results are as we expect, though with less accuracy than for the smooth examples we have computed, which we attribute to the inadequacy of the grid resolution and the finite-difference stencils to properly capture the discontinuities of the rotating signed distance function. Note that most other methods would suffer from this same issue; but our method gains robustness from computing the AKVF in a neighbourhood of the implicit surface.

Symmetries. Finally, our implementation does not suffer unduly from the exceptional scenario expressed in Theorem 3 leading to an underdetermined linear system. In part, this is because this rarely occurs in practice. But when it does occur, e.g. in the first frame of the articulated bar sequence shown in Figure 11 where the bar has a cylindrical symmetry, the conjugate gradient algorithm chooses the

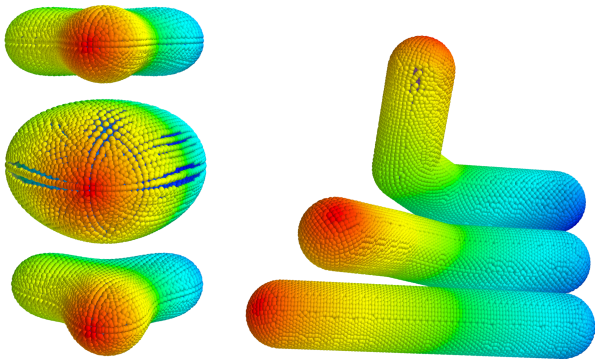


Figure 12: (left) Tracked particles for the sequence in Figure 10. A rotating camera position keeps the same lobe in view. (right) Tracked particles for the sequence in Figure 11. In both cases, particles are colored by their x-coordinate in the first frame.

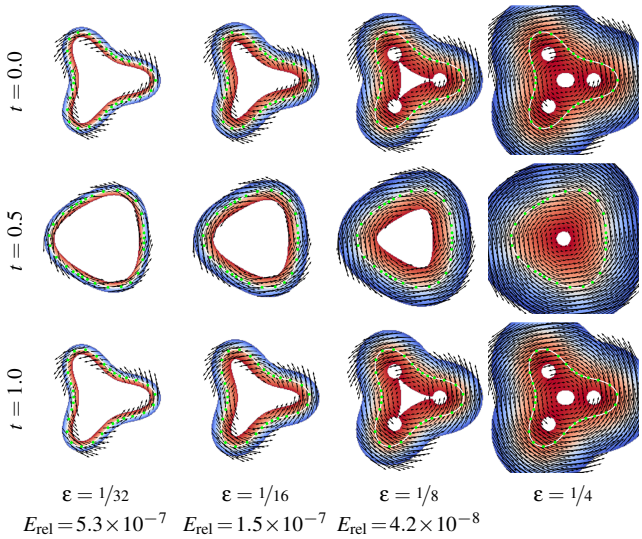


Figure 13: AKVFs tracking a global non-rigid motion of a two-dimensional shape, with varying choices of ε given as a fraction of the width of the background grid. Each experiment is marked with average tracking error for the green points relative to the tracking result using the thickest band on the right.

least-norm solution. Geometrically, this corresponds to assuming that the component of the motion in the direction of the infinitesimal cylindrical symmetry vanishes. But this may not be the case, e.g. if the bar in were rotating about its axis in the first frame. In order to capture this type of motion, a more robust solution might be to consider the AKVFs in multiple timesteps to ensure that temporal continuity is preserved. We leave this as future work.

5.4. Numerical Tests

Discretization. Table 1 tests sensitivity of the AKVF computation to discretization of time and space. Here, we compute AKVFs for rigid motions of an ellipsoid; this simple sequence has the advantage that the ground-truth AKVF is known in closed form. First, we discretize the grid $x \in [0,1]^3$ using varying densities and compute gradients of the level set function from samples, with closed-form derivatives in time. Second, we discretize time $t \in [0,1]$ using different time steps Δt, with finite differences for time derivatives and closed-form spatial gradients. Each table entry shows mean-square error of the AKVF over the narrow-band. The spatial discretization test averages this value over 10 frames, and the time discretization test averages this value over one unit of time. Overall, we see that errors are in the 0.5% range, even coarse discretizations yield $< 2\%$ error, and that these values improve as the discretization is refined.

Spacing	Rot.	Rot.+Trans.
50^3	1.99×10^{-3}	4.18×10^{-3}
60^3	1.69×10^{-3}	3.57×10^{-3}
70^3	1.49×10^{-3}	3.15×10^{-3}
80^3	1.33×10^{-3}	2.82×10^{-3}
90^3	1.21×10^{-3}	2.57×10^{-3}

Grid resolution

Δt	Rot.	Rot.+Trans.
.100	7.44×10^{-3}	1.58×10^{-2}
.050	3.31×10^{-3}	7.05×10^{-3}
.030	2.13×10^{-3}	4.53×10^{-3}
.025	1.57×10^{-3}	3.34×10^{-3}
.020	1.24×10^{-3}	2.64×10^{-3}

Time resolution

Table 1: Sensitivity to discretization in space (left) and time (right) for rigid motions; entries are in units of mean square relative error in the distance between tracked and ground-truth particle trajectories.

Timing. The supplementary document shows detailed timing information. The most expensive steps are assembling the stiffness matrix for the narrow band at a given time; and solving the equation (12). Generally, the total processor time per time step scales roughly linearly with the number of vertices in the narrow band and equals 0.1 seconds (2D) and 2.5 seconds (3D) at the resolutions we typically consider. The reason for the large discrepancy is due to the difference in sparsity of the stiffness matrices.

5.5. Tracking Implicit Surface Deformations

In this section we consider surface deformations obtained from geometric flows, meaning that the animated level set data is generated algorithmically once the initial surface is given.

Implicit surface interpolation. Figure 16 shows an example of AKVFs and particle tracking for a shape interpolation example appearing in [SDGP*15], which algorithmically generates indicator functions for the intermediate shapes based on optimal mass transport. The shape interpolation method in that paper is highly non-rigid, yet our method stably tracks particles while maintaining a smooth distribution. We extract signed distance functions numerically from the shape indicator functions using the fast marching algorithm and compute their derivatives via finite differences. We use these in our AKVF computations.

Mean curvature flow. As a second example of AKVF tracking of shapes defined using fully-implicit grid-based algorithms, Figure 17 shows AKVFs and tracked points for a surface undergoing volume-preserving mean curvature flow, using the level set algorithm presented in [Sme03] which generates the level set functions for the intermediate shapes by solving a discretized Hamilton-Jacobi equation. Once again, the tracked points smoothly move from one frame to the next and maintain their distribution, despite the fact that the shape deforms highly non-rigidly.

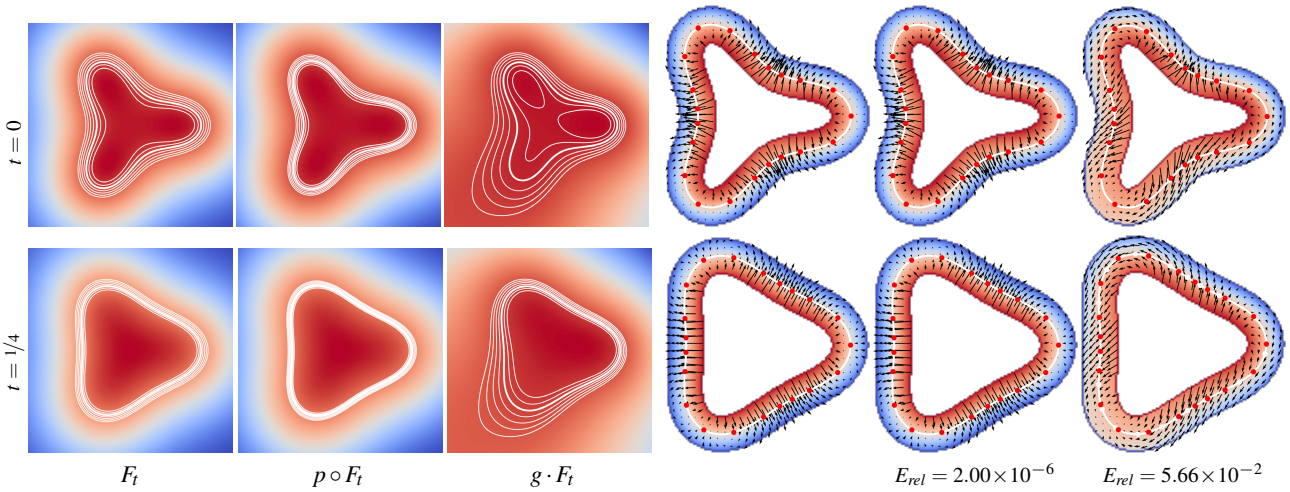


Figure 14: Tracking different level set functions for the same evolving geometry. Level sets at two frames are shown for a 2D non-rigid deformation sequence; the basic sequence (column 1) is modified by composing with a polynomial (column 2) and scaling by a function (column 3). The remaining columns show corresponding tracked points and AKVFs as well as tracking error relative to the basic sequence.

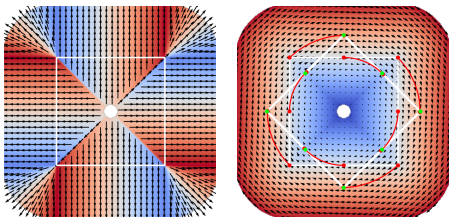


Figure 15: (left) Time derivative and gradient of the signed distance function of the square. (right) Tracking results and vector fields at the final frame for a square rotating through 45°.

5.6. Adaptations for Changing Topology

We present strategies for dealing with ambiguities that occur when a deforming level set function undergoes a change of topology, as discussed in §3.2 and §4.3. Based on geometric cues alone, particles in a neighborhood where such a change is taking place have no information regarding which component of the interface they should follow. The desired behavior of these particles is application dependent and so we present two approaches that can be taken.

The geometry in Figure 18 illustrates our approaches. We consider two blobs in 2D merging and separating, which we show on the left side of each panel together with AKVFs and particle locations. The right side of each panel shows the space-time surface swept out be the geometry as time advances (the vertical direction represents time); the time slices of interest are indicated by the presence of particles. Note the AKVFs are well-behaved and reasonable.

In the left panel of Figure 18, we observe a failure of the projection strategy as one particle gets stranded in the hole created around the point where ∇F_t vanishes as it touches the level set. This particle simply stops moving — and thus detaches from the geometry. One approach is thus to permanently delete such particles when these degeneracies occur. This is appropriate when two implicit surfaces merge and surface area is permanently removed. On the other hand, if the topological changes under consideration are temporary, it may be desirable to recover particles when the topology change is undone. In this situation we propose to maintain a simple mass-spring system, similar to [SS11] so that particles naturally gravitate back to their original surfaces. Having such a mesh is also convenient for re-establishing the original particle density, as particles tend to clump when undergoing very far from rigid deformations typical of most topology changes. We show this in the right panel of Figure 18.

6. Discussion and Conclusion

Even the simple tests at the beginning of §5 are sufficient to show the advantages of our technique over existing work. Primarily, AKVFs are ideally suited to capturing *all* classes of rigid motion rather than translational motion, and furthermore as surfaces deform elastically our method continues to provide a reasonable and intuitive model for tracking points. Although our algorithm requires the solution of a PDE rather than application of a per-point formula, in practice this step provides resilience to noisy and sampled level set functions that may not be known in closed-form.

Additional extensions of this technique can improve its efficiency or stability. For instance, the choice of the parameter ε could be carried out automatically and varied along the domain depending on the band size needed to resolve surface features and motion. An even more sophisticated implementation might employ an adaptive octree-style data structure that adds degrees of freedom to our systems of equations only in areas of the domain where more detail is needed for accurate tracking. Multigrid methods and/or preconditioned iterative solvers may also assist in solving the system of equations for AKVFs on a grid more efficiently.

Even without these improvements, AKVF-based tracking remains an efficient and accurate alternative to existing techniques for determining the velocity of an implicit surface that can be easily incorporated into existing pipelines for implicitly defined geometries.

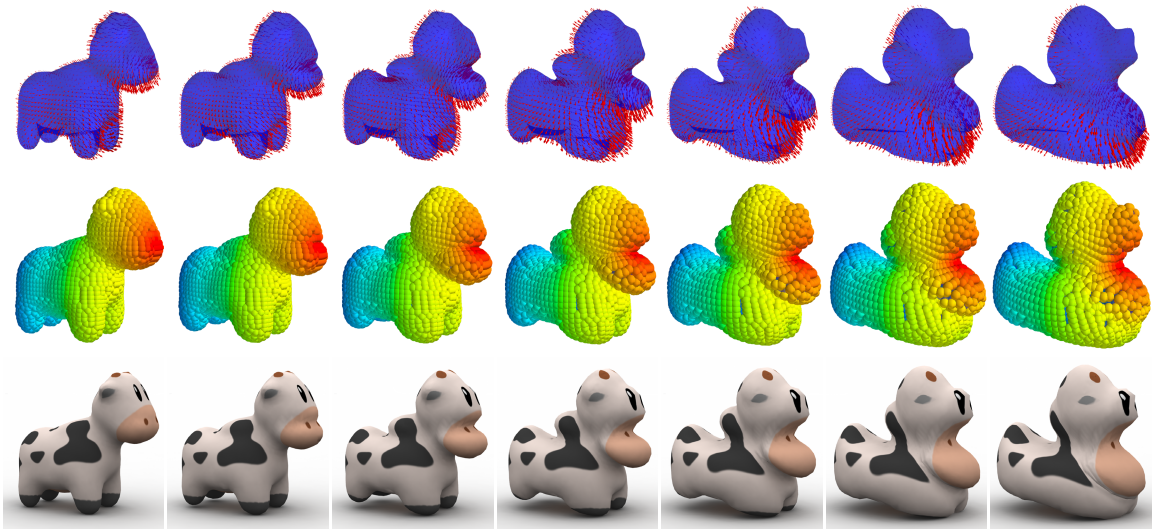


Figure 16: AKVFs (top), tracked particles (middle), and tracked texture (bottom) for the implicit shape interpolation example from [SDGP*15].

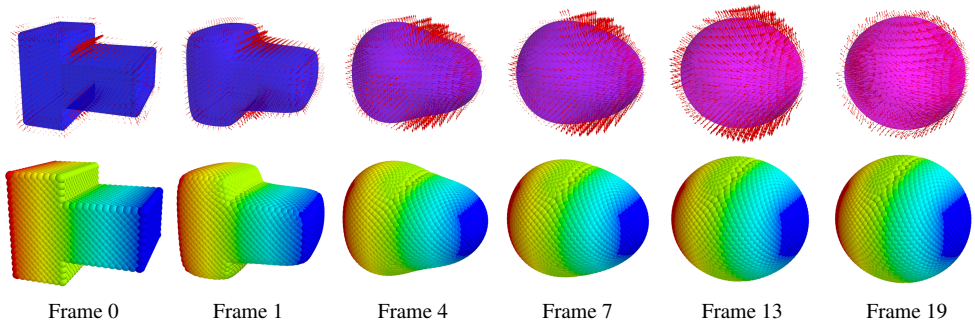


Figure 17: AKVFs (top) and tracked particles (bottom) for a shape undergoing implicit volume-preserving mean curvature flow from [Sme03].

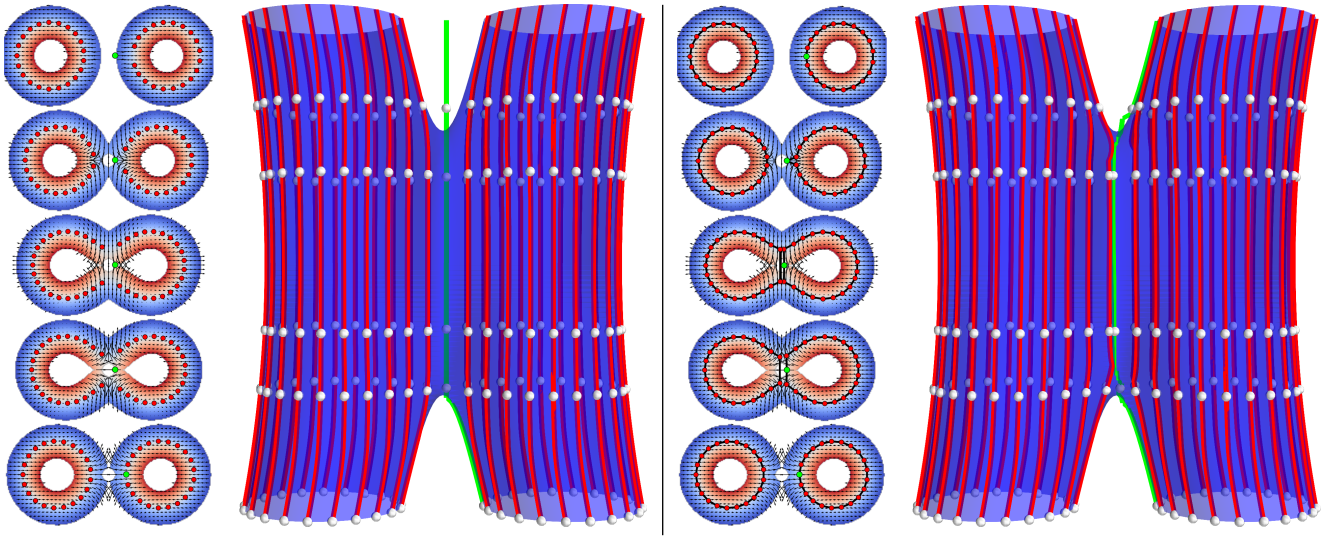


Figure 18: Strategies for tracking points. (left) Merging blobs and a lost particle. (right) Mesh strategy. See discussion in §5.6.

Acknowledgements. J. Solomon acknowledges the support of the NSF Mathematical Sciences Postdoctoral Research Fellowship (award number 1502435). We thank R. Schmidt and J. Stam of Autodesk Research for insightful conversations. Additionally we thank E. Fiume of the University of Toronto for his support and insight.

References

[ABCCO13] AZENCOT O., BEN-CHEN M., CHAZAL F., OVSJANIKOV M.: An operator approach to tangent vector field processing. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 73–82. 2

[AJT04] ALLAIRE G., JOUVE F., TOADER A.-M.: Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics 194*, 1 (2004), 363–393. 3

[AOCBC15] AZENCOT O., OVSJANIKOV M., CHAZAL F., BEN-CHEN M.: Discrete derivatives of vector fields on surfaces–an operator approach. *ACM Transactions on Graphics (TOG) 34*, 3 (2015), 29. 2

[AS99] ADALSTEINSSON D., SETHIAN J. A.: The fast construction of extension velocities in level set methods. *Journal of Computational Physics 148*, 1 (1999), 2–22. 5

[BCBSG10] BEN-CHEN M., BUTSCHER A., SOLOMON J., GUIBAS L.: On discrete Killing vector fields and patterns on surfaces. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1701–1711. 1, 2, 4

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (July 1982), 235–256. 1

[BN07] BOUTHORS A., NESME M.: Twinned meshes for dynamic triangulation of implicit surfaces. In *Proceedings of Graphics Interface 2007* (2007), ACM, pp. 3–9. 2

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. Taylor & Francis, 2008. 1

[CW98] CHEN Y.-Z., WU L.-C.: *Second order elliptic equations and elliptic systems*, vol. 174. American Mathematical Soc., 1998. 4

[dC92] DO CARMO M.: *Differential Geometry*. Prentice-Hall, 1992. 3

[DYT05] DINH H. Q., YEZZI A., TURK G.: Texture transfer during shape transformation. *ACM Transactions on Graphics (TOG) 24*, 2 (2005), 289–310. 2

[EFFM02] ENRIGHT D., FEDKIW R., FERZIGER J., MITCHELL I.: A hybrid particle level set method for improved interface capturing. *Journal of Computational physics 183*, 1 (2002), 83–116. 2

[FMM13] FUJISAWA M., MANDACHI Y., MIURA K. T.: Calculation of velocity on an implicit surface by curvature invariance. *Journal of Information Processing 21*, 4 (2013), 674–680. 2, 7

[Fre42] FREUDENTHAL H.: Simplizialzerlegung von beschränkter flachheit. *Ann. of Math. 43* (1942), 580–582. 5

[HLW06] HAIRER E., LUBICH C., WANNER G.: *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, vol. 31. Springer Science & Business Media, 2006. 6

[JLJ98] JOST J., LI-JOST X.: *Calculus of variations*, vol. 64. Cambridge University Press, 1998. 4

[Max83] MAX N.: Computer representation of molecular surfaces. *IEEE Comput. Graph. Appl. 3*, 5 (1983), 21–29. 1

[MUM14] MANDACHI Y., USUKI S., MIURA K. T.: Velocity calculation of 2d geometric objects by use of surface interpolation in 3d. *Journal of Advanced Mechanical Design, Systems, and Manufacturing 8*, 2 (2014), JAMDSM0012–JAMDSM0012. 2

[Mun91] MUNKRES J.: *Analysis on manifolds, ser. Advanced Book Classics*. Addison-Wesley Pub. Co., Advanced Book Program, 1991. 3

[Mus13] MUSETH K.: Vdb: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG) 32*, 3 (2013), 27. 5

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 30. 2

[OF02] OSHER S., FEDKIW R.: *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer, 2002. 2, 3

[OS88] OSHER S., SETHIAN J. A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics 79*, 1 (1988), 12–49. 2

[PHKF03] PONS J.-P., HERMOSILLO G., KERIVEN R., FAUGERAS O.: How to deal with point correspondences and tangential velocities in the level set framework. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (2003), IEEE, pp. 894–899. 2

[PT92] PAYNE B. A., TOGA A. W.: Distance field manipulation of surface models. *IEEE Comput. Graph. Appl. 12*, 1 (Jan. 1992), 65–71. 1

[SBCBG11a] SOLOMON J., BEN-CHEN M., BUTSCHER A., GUIBAS L.: As-Killing-as-possible vector fields for planar deformation. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1543–1552. 1, 2, 4, 6

[SBCBG11b] SOLOMON J., BEN-CHEN M., BUTSCHER A., GUIBAS L.: Discovery of intrinsic primitives on triangle meshes. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 365–374. 1, 2, 4

[SDGP*15] SOLOMON J., DE GOES F., PEYRÉ G., CUTURI M., BUTSCHER A., NGUYEN A., DU T., GUIBAS L.: Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 66. 10, 12

[Set99] SETHIAN J. A.: *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3. Cambridge university press, 1999. 5

[Sha02] SHAPIRO V.: Solid modeling. In *Handbook of Computer Aided Geometric Design*, Farin G., Hoschek J., Kim M., (Eds.). Elsevier, 2002. 1

[Sme03] SMEREKA P.: Semi-implicit level set methods for curvature and surface diffusion motion. *Journal of Scientific Computing 19*, 1 (2003), 439–456. 10, 12

[SS11] STAM J., SCHMIDT R.: On the velocity of an implicit surface. *ACM Transactions on Graphics (TOG) 30*, 3 (2011), 21. 1, 2, 7, 8, 11

[VGB*14] VAILLANT R., GUENNEBAUD G., BARTHE L., WYVILL B., CANI M.-P.: Robust iso-surface tracking for interactive character skinning. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 189. 2

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 269–277. 2

[WWG03] WANG M. Y., WANG X., GUO D.: A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering 192*, 1 (2003), 227–246. 3